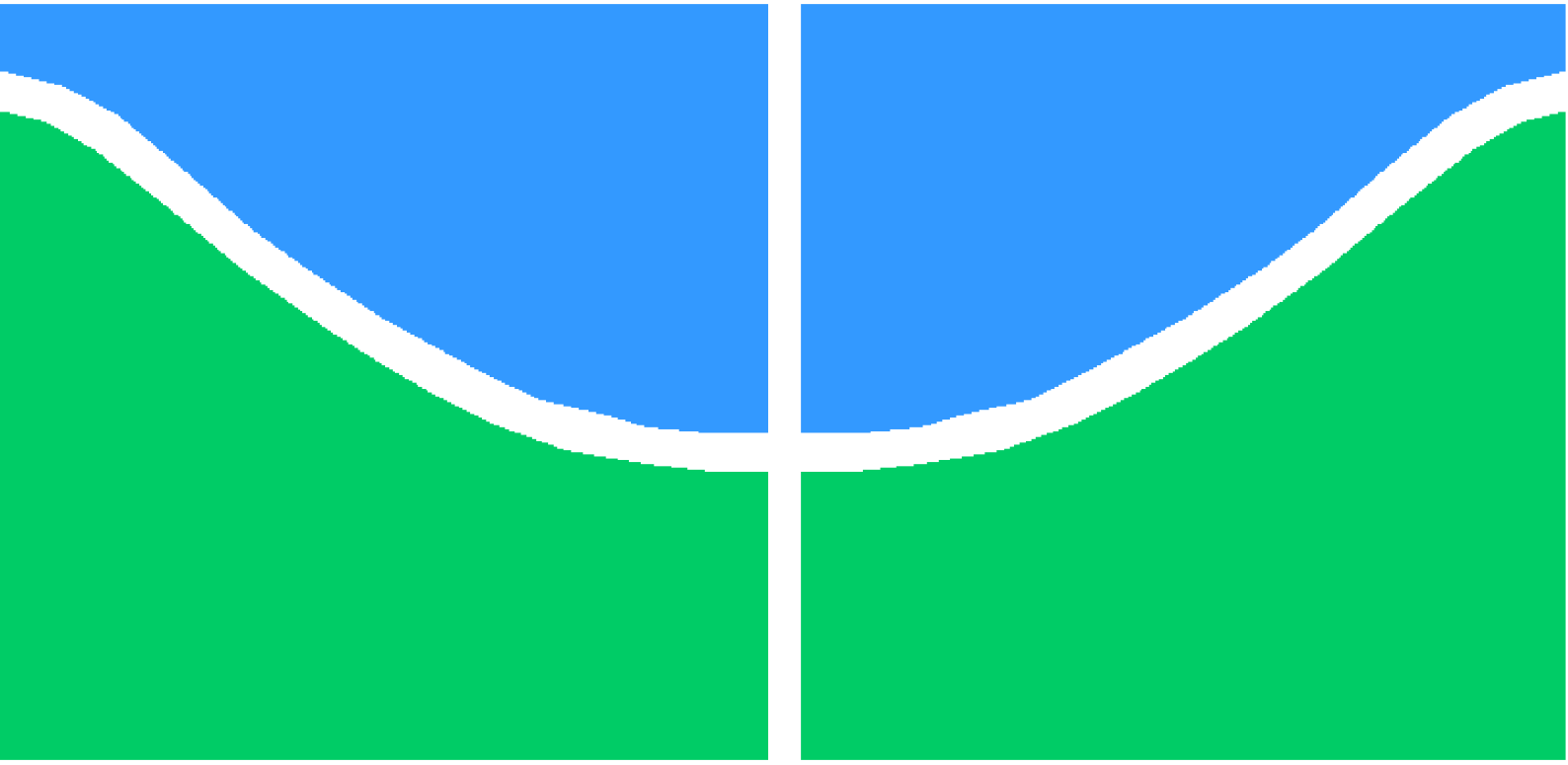


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Hypervideos: Um Sistema Adaptativo de Vídeos Interativos

Autor: Arthur Jahn Sturzbecher
Orientador: Prof. Dr. Ricardo Ramos Fragelli

Brasília, DF
2015

Arthur Jahn Sturzbecher

Hypervideos: Um Sistema Adaptativo de Vídeos Interativos

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Ricardo Ramos Fragelli

Brasília, DF

2015

Arthur Jahn Sturzbecher
Hypervideos: Um Sistema Adaptativo
de Vídeos Interativos/ Arthur Jahn Sturzbecher. – Brasília, DF, 2015-
82 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Ricardo Ramos Fragelli

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Quantização de Redes por Nodos. 2. Vídeos Interativos. 3. Navegação Adaptativa. 4. Hipermídias Adaptativas. 5. Aprendizado Multimídia. I. Prof. Dr. Ricardo Ramos Fragelli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Hypervideos: Um Sistema Adaptativo
de Vídeos Interativos

CDU

Arthur Jahn Sturzbecher

Hypervideos: Um Sistema Adaptativo de Vídeos Interativos

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, Dezembro de 2015:

Prof. Dr. Ricardo Ramos Fragelli
Orientador

Prof. Dr. Edson Alves da Costa Júnior
Convidado 1

Prof. Dr. Fábio Macêdo Mendes
Convidado 2

Brasília, DF
2015

Dedico este trabalho à minha mãe Fátima,
ao meu pai Ênio, às minhas irmãs e
namorada pelo apoio e compreensão.

Agradecimentos

Agradeço à Universidade de Brasília por me dar a oportunidade de estudar e participar do mundo acadêmico, ao REUNI, por me garantir algum auxílio durante grande parte dos meus estudos, aos professores que se fizeram presentes na minha graduação e me mostraram o norte em momentos difíceis.

Ao professor Ricardo Ramos Fragelli, que me ajudou a traçar meu caminho na faculdade e me serviu de exemplo em muitos momentos de dúvida. Ao professor Hilmer Rodrigues Neri que me mostrou muito do que conheço sobre Engenharia de Software e acreditou em mim quando até eu desacreditava.

Agradeço à minha mãe, Fátima Lúcia, pela confiança e motivação mesmo quando eu não me fazia merecedor, por me dar a oportunidade de sair de casa e viver uma época muito importante de responsabilidade e crescimento fundamentais para minha formação, não apenas acadêmica, mas também humana.

Ao meu pai, Ênio, pelo chimarrão reconfortante nos fins de tarde cinzentos das terças no Gama, pelas conversas longas e animadoras.

À minha namorada, Anna Maria, pelo amor, apoio e pelas noites de sono perdidas para que este trabalho fosse concluído. Às minhas irmãs, Agnes e Clarissa por compreenderem meu distanciamento.

Agradeço a todos os meus amigos que me apoiaram e me ajudaram durante esses anos de graduação, em especial ao Ítalo, ao Áulus, ao Levino, a Ana Paula, ao José Alisson, ao Tiago, Ramon e Matheus que me garantiram ótimas risadas e muito estudo! Agradeço a equipe Tomaz, Thaiane e Maxwell pelo esforço, admiração e apoio sempre!

Agradeço a Deus, pela oportunidade de pensar e existir.

"Nesta caminhada de cinco a seis horas (ou seriam anos),
Armazenei infinitas tonalidades e cores
Que permanecem vivas e se estendem feito sombras adocicadas
Quando resolvo passear dentro de mim! "
(Ênio Rudi Sturzbecher, fragmento de Somando Perdas)

Resumo

Segundo o Instituto Nacional de Estudos e Pesquisas Educacionais (INEP), mais de 90 por cento dos ingressantes nos cursos de engenharias não chegam a se formar. Dentre os fatores que influenciam essa estatística, está o desnível entre os estudantes e a desmotivação devido ao grande número de reprovações. Buscando colaborar com a diminuição dos desistentes, este trabalho aplica teorias sobre sistemas adaptativos e de objetos de aprendizagem na construção de um sistema adaptativo que utiliza a Quantização de Redes por Nodos(QRN) em um ambiente de vídeos interativos, que se comporta de formas distintas segundo diferentes perfis de estudantes e serve como um material de auxílio aos professores fora da sala de aula. Além disso, um aspecto importante abordado neste trabalho é o estudo das Hipermídias Adaptativas e, em especial, dos vídeos interativos, recursos que permitem a interação entre o aprendiz e o material de ensino, condição que corrobora para a ocorrência de aprendizagem significativa, confirmando a possibilidade da utilização da QRN como algoritmo para quantização para redes hipermídia.

Palavras-chaves: Quantização de Redes por Nodos. Vídeos Interativos. Navegação Adaptativa. Hipermídias Adaptativas. Aprendizado Multimídia.

Abstract

According to the Brazilian National Institute for Educational Studies and Research (INEP), more than 90 percent of freshmen in engineering courses fail to graduate. Among the factors that influence this statistic is the gap between students and discouragement due to the large number of failures. Seeking to collaborate with decreasing dropouts, this study applies theories of adaptive systems and learning objects for building an adaptive system that uses the Network's Quantization by Nodes (QRN) in an environment of interactive videos and behaves in different ways according to different profiles of students and serving as a material aid to teachers outside of the classroom. In addition, an important aspect addressed in this work is the study of the Adaptive hypermedia and in particular interactive videos, features that allow interaction between the learner and the learning material, a condition which corroborates to the occurrence of meaningful learning, confirming the possibility of using the QRN as an algorithm for quantization of hypermedia networks.

Key-words: Network Quantization by Nodes. Interactive videos. Adaptive navigation. Adaptive hypermedia. Multimedia learning.

Lista de ilustrações

Figura 1 – Relação entre matriculados e concluintes em cursos de engenharia e construção (dados obtidos do INEP para a grande área Engenharia, Construção e Produção). 27
Figura 2 – Teoria cognitiva para a aprendizagem multimídia. 31
Figura 3 – Operações de fecho. 36
Figura 4 – Processo de quantização da rede. 41
Figura 5 – Model - View - Controller. 44
Figura 6 – transferências e requisições JavaScript. 45
Figura 7 – Estilos Aquiteturais MVC. 46
Figura 8 – Defeito × Erro × Falha. 47
Figura 9 – Diagrama do vídeo interativo proposto. 51
Figura 10 – Adaptação do hypervideo por meio da ocultação dos nodos N2 e N6. . 52
Figura 11 – Diagrama da estrutura do curso proposto. 52
Figura 12 – Diagrama sobre informações referentes a relação usuário × curso. . . . 53
Figura 13 – Arquitetura Interna do *Framework Meteor*. 55
Figura 14 – Evolução da arquitetura da aplicação 56
Figura 15 – Diagrama de pacotes da arquitetura. 57
Figura 16 – Visualização dos resultados de teste da aplicação. 58
Figura 17 – Cobertura para componentes *Polymer*. 59
Figura 18 – Ferramenta para monitoramento de recursos utilizados pela aplicação. . 60
Figura 19 – Ferramenta de Integração Contínua para o projeto Hypervideos. 61
Figura 20 – Status da última construção do projeto no repositório de código. 61
Figura 21 – Funções do módulo de autoria no nível conceitual. 64
Figura 22 – Funções do módulo de autoria no nível de construção do hypervideo. . 65
Figura 23 – Página de busca por cursos. 66
Figura 24 – Módulo de visualização adaptativa. 66
Figura 25 – Componente de apresentação dos hypervideos indicados. 67

Lista de tabelas

Tabela 1 – Tempo para implementação da funcionalidade. 54

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CBL	*Challenge Based Learning* ou Aprendizagem Orientada a Desafios.
DDP	*Distributed Data Protocol*
FBI	*Federal Bureau of Investigation*
GPL	*General Public License*
HA	Hipermídias Adaptativas.
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais.
MVC	*Model - View - Controller* ou Modelo - Visão - Controle
NPR	*National Public Radio*
OA	Objetos de Aprendizagem.
PBL	*Problem Based Learning* ou Aprendizagem Orientada a Problemas.
QRN	Quantização de Redes por Nodos.
SHA	Sistemas de Hipermídias Adaptativas.
TCC	Trabalho de Conclusão de Curso.
VILS	*Video Interactive Learning System*

Lista de símbolos

A_i	Avaliação de um Nodo i
A_{0M}	Primeira avaliação sugerida para avaliação indireta
C_i	Confiabilidade da avaliação de um Nodo i
d^2	Variável de decisão para a Rede Quantizada
ε	Avaliação considerada excelente (expertise)
k_a	Grau de liberdade para variação da avaliação indireta
k_1	Coeficiente de quantização para blocos coesos
k_2	Coeficiente de quantização para direção e expertise
k_3	Coeficiente de quantização para avaliação e confiabilidade
n_a	Número de avaliações realizadas
N_a	Nodo atual que representa principal variável para quantização da rede
N_i	Nodo i da rede
n_j	Número de blocos coesos que um nodo contém
p	Coeficiente de importância de um bloco coeso
p_b	Coeficiente do percentual de expertise satisfatório
v	Incremento dos nodos segundo a regra k_2

Sumário

Introdução 25

1 REFERENCIAL TEÓRICO 29
1.1 Teorias de Aprendizagem 29
1.2 Princípios do Aprendizado Multimídia 31
1.2.1 Vídeos Interativos 32
1.3 Sistemas de Hipermídias Adaptativas 34
1.3.1 Adaptação de Conteúdo 34
1.3.2 Adaptação de Navegação 35
1.4 Quantização de Redes por Nodos 36
1.4.1 Blocos Coesos e Hierarquia de Blocos 37
1.4.2 Avaliação e Confiabilidade 38
1.4.3 Influência Temporal 39
1.4.4 Quantização das Ligações entre Nodos 40
1.4.5 Rede Behaviorista Quantizada 41
1.4.6 Rede Ausubeliana Quantizada 43
1.5 Arquitetura de software 43
1.6 Testes de Software 46
1.7 Metodologia *Scrum* 48

2 DESENVOLVIMENTO 51
2.1 Modelagem do Domínio 51
2.2 Suporte Tecnológico 53
2.3 Arquitetura do Sistema 55
2.4 Testes 58
2.5 Gerência de Configuração 59
2.6 Aspectos Gerais 61

3 SISTEMA ADAPTATIVO DE VÍDEOS INTERATIVOS 63
3.1 Módulo de Autoria de Cursos 63
3.2 Módulo de Visualização Adaptativa 65
3.3 Mecanismo de Quantização por meio da QRN 65

Considerações Finais 69

REFERÊNCIAS 73

APÊNDICES 79

APÊNDICE A – SUPORTE TECNOLÓGICO 81
A.1 Ferramentas de Suporte ao Desenvolvimento 81
A.2 Ferramentas para Ambiente de Integração contínua 82

Introdução

Este capítulo apresenta a contextualização, o problema de pesquisa, a justificativa, os objetivos, a metodologia de pesquisa utilizada e a organização deste trabalho.

Contextualização

O processo de ensino aprendizagem tem sofrido grandes mudanças desde as transformações científicas na segunda metade do século XX. O surgimento de pesquisas sobre o aprendizado e novas formas de ensino se deu naturalmente, motivado pelo desenvolvimento do pensamento sistêmico e as novas tendências de produção industrial enxuta, que exigiam trabalhos científicos e novas propostas para o sistema educacional, já que a forma programada e linear de ensino utilizado após a primeira revolução industrial não se adequava mais às necessidades contemporâneas de raciocínio e tomadas de decisão requeridas do profissional intelectual (OLIVEIRA, 2009; FRIGOTTO, 1989).

Por volta de 1950 aconteceram as primeiras tentativas de integrar a computação ao processo de ensino. Nesse período surgiram os primeiros sistemas direcionados para a educação mediada por computador, chamados Instrução Assistida por Computador, ou *Computer Assisted Instruction* (CAI). Esses sistemas tinham como pressuposto teórico o Behaviorismo, em que a instrução era feita de forma linear e subdividida em diversos módulos sequenciais. Ou seja, a instrução era baseada em uma modelagem de estímulos previamente definidos (GIRAFFA, 1995; VICARI; GIRAFFA, 2003).

Com a popularização da internet, por volta de 1990, surgiram os primeiros sistemas educativos para a web. Com base em teorias de hipermídias adaptativas (BRUSILOVSKY, 1996) e de tutoria inteligente (BRUSILOVSKY, 1994), foram criados novos ambientes educacionais adaptativos e inteligentes que atuavam segundo modelos de usuário, traçados de acordo com o uso do sistema.

Nesse mesmo período apareceram as primeiras pesquisas em sistemas de vídeos interativos que, mesmo com o uso de vídeo cassete e um monitor, permitiam maior controle e interação do aprendiz com o material de aprendizado, o que contribui para uma aprendizagem mais efetiva (ZHANG, 2005). Na época, foi possível verificar que a atividade era promissora mesmo na fase inicial de aplicação (GAUDREAU; CHAN, 1984).

Já nos trabalhos mais recentes, há uma tendência para a utilização de sistemas tutores inteligentes combinados com as teorias de hipermídias adaptativas para uma melhor aprendizagem (FRAGELLI, 2010). Fragelli discorre sobre uma nova abordagem para quantização de redes hipermídia que leva em consideração apenas as informações dos

nodos, a Quantização de Redes por Nodos (QRN), e integra as teorias de hipermídias adaptativas, sistemas tutores inteligentes e objetos de aprendizagem (OA), que no caso específico deste trabalho, se configuram como vídeos interativos.

Problema de Pesquisa

Com base no contexto apresentado, este trabalho pretende verificar se é possível desenvolver uma navegação global adaptativa para um sistema *web* de vídeos interativos utilizando a Quantização de Redes por Nodos.

Justificativa

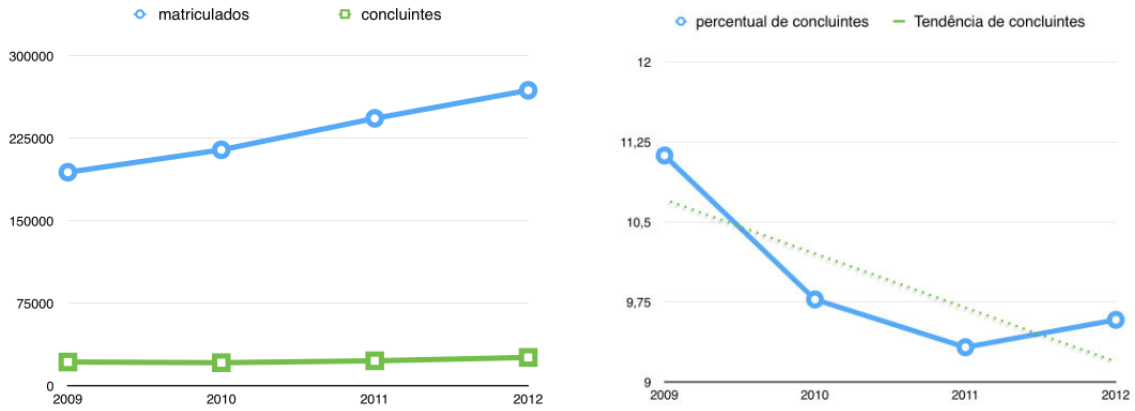
Apesar dos esforços realizados no desenvolvimento de sistemas inteligentes e adaptativos no processo de ensino, poucos foram os trabalhos que se preocuparam com a implantação e manutenção prática da tecnologia proposta, o que reforça a afirmação de que as linhas de pesquisa estão mais voltadas para o campo técnico do que para o âmbito da aprendizagem. Além disso, a QRN ainda não possui validação em um ambiente real de uso, sendo avaliada por especialistas na área com análises em diferentes redes hipermídias e com diferentes perfis de usuário (FRAGELLI, 2010).

Outro fator a ser analisado é que os ambientes virtuais de aprendizagem exigem um grande número de OAs para serem efetivos, já que para alcançar o público de estudantes, tem-se a necessidade de produzir materiais mais elaborados e complexos segundo as diferentes necessidades cognitivas e perfis dos usuários. Para um professor-autor, essa característica desmotiva a produção dos materiais e corrobora a para a inutilização do sistema (FRAGELLI, 2010). Dessa forma, seria uma boa medida se o sistema proposto contemplasse mecanismos facilitadores para a produção de recursos educacionais.

Ademais, segundo a Sinopse de Educação Superior publicada pelo Instituto Nacional de Estudos e Pesquisas Educacionais (INEP), o índice de desistência dos cursos na área de Engenharia, Produção e Construção nas instituições públicas chega a mais de 90% dos matriculados (Fig. 1), o que indica uma grande desmotivação por parte dos alunos que pode ser explicada, dentre outros fatores, pelo alto nível de reprovações e pelo desnível entre os estudantes (SILVA, 2005).

Com base nesses fatores, o professor tem que alcançar um equilíbrio entre o mínimo e o razoável a ser ensinado, o que dificulta o aprofundamento em conteúdos mais elaborados e desafiadores, tornando frustrante o papel de educador, refletido em aulas monótonas e clássicas para um grupo de estudantes que vive o dinamismo tecnológico (FRAGELLI, 2010).

Dessa forma, é necessário desenvolver sistemas de ensino que possibilitem uma aprendizagem adaptativa. Contudo, é também importante motivar os professores a fazerem parte do processo de construção dos materiais que alimentem tais sistemas.



(a) total de matriculados e concluintes.

(b) percentual de concluintes.

Figura 1 – Relação entre matriculados e concluintes em cursos de engenharia e construção (dados obtidos do INEP para a grande área Engenharia, Construção e Produção).

Objetivos

O objetivo geral deste trabalho é desenvolver um sistema de vídeos educacionais interativos com navegação global adaptativa utilizando a QRN. Para tal, tem-se os seguintes objetivos específicos:

- Desenvolver um módulo de autoria de vídeos interativos que permita a criação dos objetos de aprendizagem, de modo a incentivar professores a construírem esses materiais.
- Desenvolver um módulo de visualização dos vídeos interativos que adapte o conteúdo para os diferentes perfis de estudantes.
- Desenvolver um mecanismo de navegação global adaptativa com a QRN para garantir a adaptação da navegação para diferentes perfis de estudantes.

Metodologia

Levando em consideração os objetivos deste TCC, é utilizada a pesquisa exploratória com o intuito de compreender as linhas de pesquisa relacionadas aos sistemas inteligentes e adaptativos para o ensino, tendo como base os estudos em QRN, Sistemas de Hipermídias Adaptativas (SHA) e Vídeos Interativos para o desenvolvimento do sistema proposto.

Tendo isso em mente, e considerando também o âmbito da engenharia de software, o processo de desenvolvimento deste trabalho está fundamentado nas metodologias ágeis, mais especificamente as práticas do *Scrum*. Já em relação ao desenvolvimento do sistema, são utilizados os paradigmas de programação Orientado a Objetos e o estilo arquitetural MVC.

Inicialmente, foram identificados os artigos científicos, dados informativos sobre os cursos de engenharia e possíveis tecnologias que se alinhem ao propósito deste trabalho, possibilitando um maior conhecimento dos domínios de ensino assistido por computador e de engenharia de software aplicada a sistemas *web*.

O próximo passo foi definir o escopo do trabalho, levando em consideração as dimensões das áreas de pesquisa relacionadas ao ensino assistido por computador, e ao tempo disponível para o desenvolvimento, além de serem definidas e estudadas também as tecnologias mais adequadas segundo as características do sistema.

Em paralelo a essas atividades, foi feito o refinamento da proposta para melhor se alinhar aos conceitos estudados no referencial teórico e foi feita a documentação do trabalho realizado para permitir, por fim, a escrita deste trabalho e o desenvolvimento do sistema.

Organização do Trabalho

Este trabalho está subdividido em três capítulos. No Capítulo 1 é apresentado o referencial teórico utilizado nesta pesquisa, contendo um subtópico sobre os estilos de aprendizagem estudados para a proposta das redes hipermídias; posteriormente, um subtópico sobre os Princípios do Aprendizado Multimídia, exemplificando as heurísticas aplicadas na modelagem dos vídeos e como elas interferem no processo de ensino e aprendizagem; outro subtópico sobre Sistemas de Hipermídias Adaptativas e um subtópico sobre a QRN e as teorias que a fundamentam. É apresentado também o contexto da engenharia de software, com subtópicos sobre arquitetura de software, testes e metodologia *Scrum*.

No Capítulo 2 é apresentado o desenvolvimento deste TCC, a modelagem do domínio do sistema, a arquitetura proposta e os módulos de autoria de materiais e de visualização adaptativa, bem como uma visão geral das estratégias de teste e desenvolvimento utilizadas na contrução do software. O Capítulo 3 apresenta os resultados obtidos com o desenvolvimento da plataforma, sendo subdividido em três tópicos: módulo de autoria de cursos, módulo de visualização adaptativa e mecanismo de quantização por meio da QRN. Finalmente, se discorre sobre as considerações resultantes deste estudo, o estado atual do desenvolvimento e as perspectivas de trabalhos futuros.

1 Referencial Teórico

1.1 Teorias de Aprendizagem

Para qualquer profissional na área de ensino e aprendizagem é necessário o estudo das teorias de aprendizagem. No caso específico deste trabalho foi utilizada a teoria Cognitivista, que serviu como suporte teórico para a modelagem das redes de vídeos interativos utilizadas no sistema.

Segundo Moreira (1999), as teorias de aprendizagem são tentativas de interpretar sistematicamente, organizar e prever os conhecimentos relativos à aprendizagem. Hill (2002) representa as teorias como o ponto de vista sobre a abordagem de assuntos relativos ao aprendizado e a especificação de quais são as variáveis independentes, dependentes e intervenientes que possuem relevância acadêmica.

As teorias behavioristas, ou comportamentalistas, são aquelas ligadas fundamentalmente aos comportamentos observáveis e mensuráveis do indivíduo, tendo como ideia inicial e fundamental o uso de estímulos e respostas.

Concebido por John B. Watson, o behaviorismo rejeita a hipótese de que existe algo além do mundo físico. Se opondo à psicologia da época que estudava os sentimentos e pensamentos das pessoas, o behaviorismo estava centrado no que as pessoas faziam e que era observável (MOREIRA, 1999).

Segundo o behaviorismo radical, ou Skinneriano, o estímulo é o evento que afeta os sentidos do aprendiz, o reforço é o evento que aumenta a probabilidade de ocorrência de um evento que o precedeu, e as contingências do reforço são um arranjo de uma situação que favoreça a ocorrência de uma resposta que leve ao reforço. O comportamento respondente é aquele que é eliciado involuntariamente (p.ex. contração da pupila sob incidência de luz). O comportamento operante é aquele em que o aprendiz opera sobre o meio conscientemente, agrupando a maior parte dos comportamentos humanos. Semelhantemente, são definidos os condicionamentos como operantes ou respondentes, já que para todo comportamento existe um condicionamento (FRAGELLI, 2010; SILVA, 2005).

Em termos de aplicações educacionais, Skinner acreditava que o papel do professor está muito mais ligado às contingências de reforço do que ao par estímulo-resposta. Em outras palavras, o professor deve atuar no planejamento de um contexto que aumente a probabilidade do comportamento desejável acontecer. Vários fenômenos estudados por Skinner podem ser aplicados ao processo educacional, como o a modelagem e o esmaecimento.

A modelagem, ou método das aproximações sucessivas, consiste no reforço de várias respostas intermediárias que servem como uma ponte para um comportamento desejado. No esmaecimento são utilizados diferentes estímulos em conjunto com o que se deseja alcançar, e tais estímulos são esmaecidos até que se mantenha apenas o desejável.

Apesar de trazer pontos relevantes para aplicação no ensino e aprendizagem, a posição radical e inflexível de Skinner em relação ao behaviorismo e suas convicções levaram-no a ignorar completamente a psicologia cognitiva, o que culminou no declínio do behaviorismo radical (FRAGELLI, 2010; SILVA, 2005; MOREIRA, 1999).

Por volta de 1975 as pesquisas sobre a psicologia cognitiva começaram a superar as pesquisas behavioristas em número de publicações. Em parte, essa ascensão se deve ao fato de que o behaviorismo não explicava a complexidade do comportamento humano, se restringindo a usar estímulos, reforços e respostas (ROBINS; GOSLING; CRAIK, 1999).

A cognição, ou atividade mental, representa a aquisição, o armazenamento, a transformação e aplicação do conhecimento. A abordagem cognitiva é uma orientação teórica que enfatiza o conhecimento que as pessoas possuem e seus processos mentais (MATLIN, 2004).

A teoria cognitiva de Ausubel (2000) está centrada no conceito de aprendizagem significativa, para a qual o cerne da aprendizagem está no que o aprendiz já conhece, sendo que, para o aprendizado e retenção de algo novo em sua estrutura cognitiva, é necessário que existam conceitos prévios que atuem como âncoras para esse novo conceito formado.

O conceito subsunçor, ou simplesmente subsunçor, é este conceito âncora já existente na estrutura cognitiva do indivíduo, com o qual um novo conceito aprendido interage. Além disso, quando um novo conceito é ancorado ao subsunçor, ele se torna mais desenvolvido e inclusivo, mas se a aprendizagem não ocorre com frequência em conjunto com um subsunçor, ele se torna limitado e menos desenvolvido.

Com isso, se a aprendizagem significativa necessita da existência de conceitos subsunçores, é necessário então que, em um momento inicial, ocorra uma aprendizagem mecânica, que independe de conceitos prévios. Nesse sentido, principalmente na educação infantil, a aprendizagem mecânica é o ponto inicial para a aprendizagem significativa. Com o desenvolvimento da estrutura cognitiva, os conceitos vão se tornando mais elaborados e abrangentes, permitindo maior ocorrência da aprendizagem significativa (AUSUBEL, 2000).

Segundo Ausubel (2000), é mais fácil para um indivíduo aprender significativamente termos mais amplos e inclusivos para depois captar conceitos mais específicos como uma diferenciação do todo, do que aprender as partes para se chegar ao todo. Assim, Ausubel apresenta teorias como a diferenciação progressiva, que deve ser um princípio programático do material de ensino, e a reconciliação integrativa para explorar as

similaridades e diferenças entre as ideias para se propor uma nova instrução (AUSUBEL, 2000; FRAGELLI, 2010).

Já os organizadores prévios são estruturas facilitadoras do processo de aprendizagem significativa, sendo geralmente materiais introdutórios que possuem alto nível de abstração, generalidade e inclusividade, e servem como uma ponte cognitiva entre o que o indivíduo conhece e o que se pretende aprender, promovendo uma disposição do sujeito a aprender significativamente o conteúdo (AUSUBEL, 2000; TAVARES, 2010).

Nesse sentido, os organizadores prévios são particularmente especiais para a confecção de materiais digitais e interativos de aprendizagem, e têm tido sucesso em projetos nacionais e internacionais (TAVARES, 2010; NOVAK; CAÑAS, 2006). Neste trabalho, os vídeos Interativos servem como mecanismo para utilização dessa teoria, no sentido de permitirem a construção gradual do conteúdo a ser aprendido.

1.2 Princípios do Aprendizado Multimídia

Mayer e Chandler (2001) definem os princípios do aprendizado de multimídia tendo como base a teoria da codificação dual. Paivio (1986) distingue o aprendizado por imagens e animações do aprendizado pela escuta, e afirma que o canal que processa imagens é diferente do canal que processa o som e ambos têm capacidades limitadas (TAVARES, 2010).

Além disso, Mayer afirma que para que ocorra aprendizagem significativa, ambos os canais requerem um processamento cognitivo, no qual imagens e palavras são processadas, organizadas e conectadas segundo uma ordem lógica para a formação de um conceito, como ilustrado na Figura 2 (MAYER; CHANDLER, 2001; MORENO; MAYER, 2000). Nesse sentido, alguns princípios para a construção de materiais multimídia estabelecidos pela literatura foram levados em consideração para a modelagem abordada neste trabalho, sendo eles: segmentação, pré-texto, modalidade e diferenças individuais (CLARK; MAYER, 2011; MAYER; CHANDLER, 2001; MORENO; MAYER, 2000), explicados a seguir.

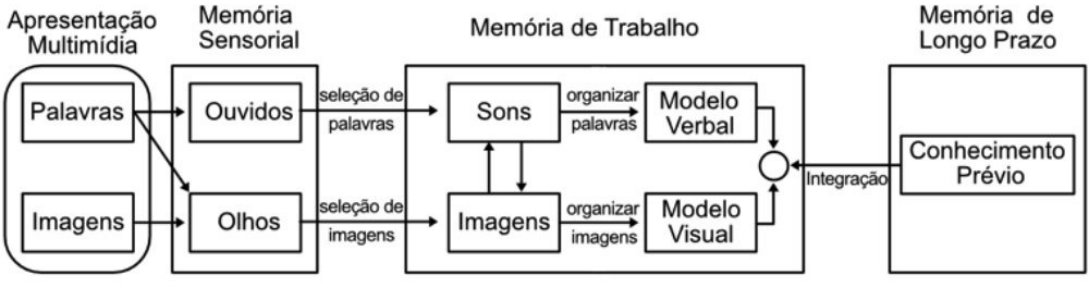


Figura 2 – Teoria cognitiva para a aprendizagem multimídia.
Fonte: (MORENO; MAYER, 2000)

O princípio da segmentação diz que o estudante aprende melhor quando um conteúdo é apresentado em vários vídeos, ou segmentos de um mesmo vídeo, do que quando é utilizado um único vídeo contínuo ininterrupto. A justificativa é que, para o aluno, o processamento cognitivo muitas vezes não acompanha o fluxo de informações contínuo para que ocorra aprendizagem significativa, o que diminui a capacidade do aprendiz de armazenar a informação passada (MAYER; CHANDLER, 2001; MORENO; MAYER, 2000).

O princípio do pré-texto diz que um estudante aprende melhor se conhecer os conceitos e tópicos que serão estudados no curso antes da apresentação do conteúdo (CLARK; MAYER, 2011). Esse princípio valoriza a utilização de tabelas ou mapas conceituais para que o estudante possa organizar os conceitos antes da definição dos mesmos, estando diretamente relacionado a teoria da apresentação "Todo-Parte", que afirma que um estudante aprende melhor quando conhece o todo antes das partes (MAYER, 2014; MAYER; CHANDLER, 2001). Esta última tem como base a teoria cognitiva das diferenciações sucessivas (AUSUBEL, 2000).

O princípio da modalidade afirma que o aprendizado é mais efetivo quando se utiliza animação e narração se comparado com o uso de animação e texto escrito. Dessa forma, devem ser valorizados discursos com representação visual e o não uso de textos explicativos simultaneamente (MAYER, 2014).

O princípio das diferenças individuais afirma que a modelagem do material multimídia afeta muito mais estudantes de níveis inferiores do que estudantes mais avançados. Esse princípio diz então que a modelagem deve ser mais direcionada para estudantes de menor nível (MAYER, 2014), ou então pode-se utilizar uma adaptação do conteúdo para adequar o material aos diferentes níveis de estudantes (FRAGELLI, 2010; BRUSILOVSKY, 1996; BRUSILOVSKY, 1994), como abordado neste trabalho.

1.2.1 Vídeos Interativos

Um Objeto de Aprendizagem (OA) pode ser definido de forma geral como qualquer material, digital ou não, que possa ser utilizado, reutilizado ou referenciado durante a aprendizagem suportada pela tecnologia (FRAGELLI, 2010). Um OA pode assumir vários formatos, desde uma apresentação textual ao mais complexo sistema de simulação. Neste trabalho, os OAs se configuram como vídeos interativos, mecanismos hipermidiáticos que possibilitam ao estudante ter o controle do material de estudo, para que este possa concentrar sua atenção nos conteúdos segundo suas expectativas e controlar o fluxo de informação, apresentando materiais de forma dinâmica e adaptativa (STURZBECHER et al., 2013).

Não é de hoje que as pesquisas indicam os aspectos vantajosos dos vídeos in-

terativos. Gaudreau e Chan (1984) conduziram um estudo que tinha como objetivo a construção do Video Interactive Learning System (VILS), que utilizava um vídeo cassete e um monitor para a implantação de um sistema de educação que oferecesse suporte aos vídeos interativos. Como resultado, foi concluído à época que o uso dessa tecnologia era promissora já na fase inicial de aplicação.

Um aspecto que diferencia os vídeos interativos dentre as formas já existentes e estabelecidas de ensino é o acesso mais rápido e simplificado ao material, fazendo com que o aprendiz construa sua própria versão de conhecimento após organizar as ideias passadas no ritmo de tempo que seja adequado a ele (STURZBECHER et al., 2013; ZHANG, 2005), podendo controlar o fluxo da apresentação e o conteúdo exibido.

Estudos recentes mostram que esses controles de tempo e conteúdo dados aos estudantes em um ambiente de vídeos são utilizados naturalmente no momento da aquisição de conhecimento, um resultado que demonstra a necessidade que o aprendiz possui de adequar o material ao seu ritmo de aprendizado e sua capacidade cognitiva (CHANDLER, 2004; MAYER; CHANDLER, 2001).

Nesse sentido, a modelagem dos vídeos interativos está diretamente relacionada ao processo cognitivo do estudante, já que cada ponto de decisão pode levar ao sucesso ou fracasso do aprendizado esperado (MAYER; CHANDLER, 2001; MORENO; MAYER, 2000). Considerando que um vídeo interativo é formado por diversos fragmentos de vídeos interligados e que essa ligação é acessada por meio da estrutura de decisão dos vídeos interativos, existem dois modelos de implementação diferenciados para essa estrutura de decisão: o modelo hipermidiático e o de multimídia (WETZEL; RADTKE; STERN, 1994).

O modelo hipermidiático de vídeos interativos tenta trazer os conceitos de hipertexto e hipermídia para o âmbito dos vídeos interativos. Segundo a teoria, as ligações existentes nos vídeos devem estar vinculadas temporalmente aos conceitos âncoras a que estão relacionadas. Isto é, caso exista um material sobre derivadas que seja de potencial interesse do estudante e, em um momento do vídeo o conceito de derivada é apresentado, nesse exato momento o link para o material deve ser disponibilizado. Dessa forma, o aprendiz pode acessar um material de determinado tema assim que este material for requerido para o ensino (WETZEL; RADTKE; STERN, 1994).

A princípio este modelo é promissor, pois garante ao estudante o acesso ao material de estudo sempre que necessário para o aprendizado. Entretanto, trabalhos recentes identificaram falhas no modelo que comprometiam a eficácia da aprendizagem, pois acarretavam na sobrecarga cognitiva do usuário com excesso de informação, requerendo seleções e avaliações além do processamento dos conteúdos ministrados (ZHANG, 2005).

As heurísticas de multimídia se afirmaram como uma boa proposta para a modelagem de vídeos interativos por simplificarem o processo de decisão e diminuírem a

carga cognitiva sobre o usuário no momento de aquisição da informação, agrupando os *hyperlinks* sempre ao final de cada vídeo.

Dessa forma, o estudante tem a liberdade de acesso aos conteúdos possivelmente interessantes, mas sem a necessidade de selecionar e avaliar os *hyperlinks* enquanto assiste a um vídeo. Essas heurísticas são exploradas por Mayer (2014), quando define os princípios do aprendizado multimídia.

1.3 Sistemas de Hipermídias Adaptativas

A hipermídia é um meio não linear de informação que, diferente do hypertexto, pode conter gráficos, áudio e vídeo, além de apenas textos e *links*. Em um sistema de hipermídias, é apresentado a todos os usuários sempre o mesmo conteúdo da página e os mesmos *links*.

As Hipermídias Adaptativas (HA) se tornaram ferramentas poderosas no desenvolvimento de sistemas de informação centrados no usuário, tendo como objetivo aumentar a funcionalidade das hipermídias por meio da personalização do conteúdo para cada indivíduo. Os Sistemas de Hipermídia Adaptativa (SHA) constroem um modelo dos objetivos, preferências e conhecimentos do usuário, e usa esse modelo como base para adaptação dos conteúdos segundo as necessidades desse usuário (BRUSILOVSKY, 1996).

Os SHA podem ser utilizados em qualquer área de aplicação em que se espera um público com diferentes objetivos e conhecimentos e que procuram acessar diferentes partes da informação contida na página hipermídia e podem usar diferentes links para navegação. Para superar esse problema, esses sistemas atuam na modelagem do perfil do usuário para adaptação do conteúdo e dos links de navegação disponibilizados para cada usuário distinto (BRUSILOVSKY, 1996; BRUSILOVSKY, 2003; FRAGELLI, 2010).

Dessa forma, é possível identificar os dois âmbitos da adaptação proposta pelos SHA, a adaptação de conteúdo e a adaptação de navegação, explicados a seguir.

1.3.1 Adaptação de Conteúdo

A ideia por trás dos métodos de adaptação de conteúdo é adequar o conteúdo da página acessada por um usuário específico aos seus conhecimentos, objetivos e características no instante em que ocorreu o acesso. Por exemplo, um estudante com maior qualificação pode receber informações mais aprofundadas e detalhadas enquanto um estudante pouco experiente pode receber explicações adicionais sobre o conteúdo (BRUSILOVSKY, 1996).

Existem diversas técnicas levantadas por Brusilovsky (1996) sobre adaptação de conteúdo encontradas em variados sistemas utilizados em meados da década de 1990, den-

tre elas: explicação adicional, explicação comparativa, explicação requerida, explicações variantes e textos condicionais.

A explicação adicional é a mais popular dentre as técnicas e seu objetivo é esconder do usuário partes do conteúdo que podem não ser relevantes para o aprendizado do indivíduo. por exemplo, explicações aprofundadas podem ser escondidas de um estudante iniciante, pois essas não serão compreendidas pelo aprendiz e podem atrapalhar o processo de aprendizagem. Ou ainda, explicações adicionais normalmente requeridas por iniciantes podem ser escondidas de um estudante com maior graduação, pois este não precisa mais dessas explicações.

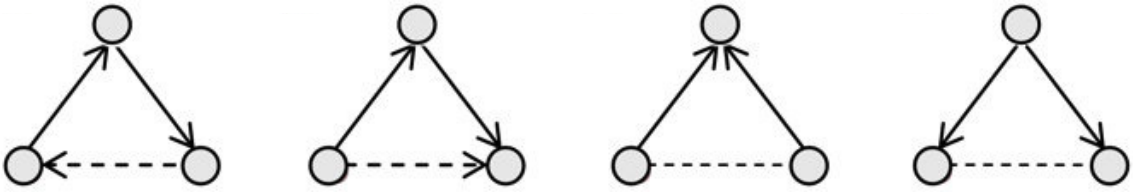
A técnica dos textos condicionais se baseia na divisão de toda a teoria sobre um conceito em vários pedaços, onde cada pedaço é associado a uma condição dependendo do nível de conhecimento do usuário, que só exibe aquele pedaço uma vez satisfeita a condição. Essa técnica é bastante flexível, podendo englobar todas as outras citadas, desde que se escolha as condições certas e os conceitos mapeados no modelo do usuário (BRUSILOVSKY, 1996).

Muito embora essas técnicas sejam em sua maioria aplicadas a sistemas textuais (BRUSILOVSKY, 1996; FRAGELLI, 2010), não é difícil adaptá-las para um sistema de vídeos interativos, pois se os subvídeos forem separados e classificados quanto a ocultabilidade e prioridade, tem-se uma boa forma de definir quais podem ser ocultados ou não na apresentação de um conteúdo. Mesmo o método das explicações variantes pode ser alcançado, com a diferença de que este requer vários vídeos diferentes sobre um mesmo tópico apresentado. Isso se dá de forma semelhante ao método dos textos condicionais, mas com o uso de vídeos.

1.3.2 Adaptação de Navegação

A adaptação de navegação tem como objetivo ajudar o usuário a encontrar seus caminhos quando imerso na rede hipermídia. Essa adaptação é feita por meio da adequação da forma como os links para outros nodos da rede são apresentados, levando em consideração o modelo de usuário construído pelo sistema, sendo muito importante para evitar que o usuário se perca na rede de informações e não seja capaz de alcançar seus objetivos. Os métodos utilizados para a navegação adaptativa são a Condução Global, a Condução Local, Suporte à Orientação Global e Suporte à Orientação Local (BRUSILOVSKY, 1996; PALAZZO, 2000).

A Orientação Global ocorre quando o usuário tem algum objetivo em um ou mais nodos da rede e o sistema o ajuda a alcançar essa informação através do caminho mais curto, com possíveis desvios. A Condução Global é o objetivo inicial dos sistemas de recuperação de informação e muito útil em sistemas de ajuda online (BRUSILOVSKY,



(a) cíclico. (b) transitivo. (c) sobrejetor. (d) sobrejetor inverso.

Figura 3 – Operações de fecho.

1996; PALAZZO, 2000).

Tendo em vista o contexto educacional e considerando que para um estudante existe apenas um objetivo global, que é o do aprendizado, os sistemas adaptativos precisam captar a dinâmica da aprendizagem do estudante com a finalidade de acelerar ou retardar tópicos segundo essa dinâmica. Essa adaptação pode ser feita por meio da quantização da rede de informações (PALAZZO, 2000), como proposto neste trabalho.

1.4 Quantização de Redes por Nodos

Antes de se especificar o conceito da QRN é necessário conhecer as teorias da quantização de redes. Palazzo (2000) define uma rede de informações como sendo uma tripla formada pelos nodos, suas ligações e o conjunto de propriedades específicas da rede. Os nodos representam unidades estruturais da rede e neste trabalho serão apresentados como vídeos interativos; as ligações são os *links* que permitem a navegação entre esses vídeos interativos.

Quantizar uma rede nada mais é do que valorar as ligações entre nodos. Dessa forma é possível saber quais nodos estão mais próximos, ou são mais relevantes entre si. Existem duas formas de se quantizar uma rede: a quantização retroativa e a proativa.

Na quantização retroativa o acesso a um *link* da rede promove um acréscimo ao valor da ligação. Esse valor atribuído a uma ligação é chamado de potencial de ativação, pois tenta aproximar por acessos anteriores o potencial que esse *link* tem de ser acessado novamente (PALAZZO, 2000).

A quantização proativa é um processo complementar, no qual são feitas inferências sobre a rede por meio das operações de fechamento, tendo como base a noção matemática de fecho e suas principais classificações: cíclico (Fig. 3a), transitivo (Fig. 3b), sobrejetor (Fig. 3c) e sobrejetor inverso (Fig. 3d) (PALAZZO, 2000; FRAGELLI, 2010). Por meio dessas inferências é possível antecipar possíveis estados futuros da rede, induzindo o surgimento de novas configurações.

Com base nessas operações, foram definidos operadores capazes de promover um auto aprendizado da rede. Bollen e Heylighen (1996) ratificaram essa utilização em uma

rede de informações distribuída com múltiplos usuários. Os operadores também foram utilizados em uma rede hipermídia na qual foi incluída a operação de degradação da rede, de modo que os acessos mais recentes tinham maior peso em favor de uma melhor navegação adaptativa (PALAZZO, 2000).

Essas teorias possuem um grande espaço de aplicação em auto organização de sistemas dinâmicos e são utilizadas em diversos campos como cibernética, biologia, termodinâmica, matemática e também evolução da linguagem.

O grande problema dessa abordagem quando aplicada a ambientes de ensino é que não há como relacionar o processo de aprendizagem de um indivíduo ao comportamento comum de usuários em uma rede, pois cada estudante possui objetivos diferentes ou capacidades cognitivas diferentes, dentre várias outras características importantes no processo de ensino e aprendizagem que são únicos para cada pessoa.

A QRN então surge como uma possibilidade de quantização que leve em consideração apenas as informações contidas nos nodos para se gerar a quantização da rede, possibilitando sua utilização em um sistema de cursos on-line (FRAGELLI, 2010).

Para se calcular o potencial de ativação entre os nodos da rede é utilizada a menor distância entre o nodo atual e os demais. No caso de uma rede unidimensional de fluxo crescente, se o nodo atual é o nodo cinco e se sua avaliação for satisfatória, o estudante será direcionado para o nodo seis; caso a avaliação seja insatisfatória, o estudante será direcionado para o nodo quatro. Isso exemplifica a utilização da menor distância para se inferir qual o melhor caminho a ser seguido, contudo ainda falta especificar os conceitos, variáveis e condições para se quantizar a rede, explorados nos tópicos a seguir.

1.4.1 Blocos Coesos e Hierarquia de Blocos

Os blocos coesos são formados por conceitos que não podem ser explicados separadamente, isto é, sempre devem ser apresentados em ordem e não podem ser ocultados individualmente. Por exemplo, a soma de vetores por componentes e a decomposição de vetores são dois conceitos dependentes, pois a decomposição de vetores é necessária para se aprender a soma de vetores por componentes. Em contrapartida, existem conceitos que abordam um mesmo assunto mas de formas diferentes, que não carecem de uma ordem definida de exibição, como a soma de vetores por componentes e pela regra do paralelogramo. Podem existir inúmeras formas de se explicar um mesmo assunto, por isso é necessário especificar uma hierarquia de blocos, a fim de se especificar um índice de importância para cada uma delas (FRAGELLI, 2010).

Essas teorias são de importância maior quando se utiliza um Sistema Tutor Inteligente, pois este atua na seleção de conteúdos para a confecção do material a ser exibido pelo sistema (FRAGELLI, 2010), exigindo uma organização dos conceitos dependentes.

Já no caso deste trabalho, o material estará pronto a partir do momento que for criado pelo professor autor, não requerendo uma seleção prévia. Entretanto, os blocos coesos podem ser explorados no momento da confecção do material, por meio da criação das ligações entre os nodos, o que dá preferência aos nodos com ligações diretas.

A hierarquia de blocos é também abordada no momento da confecção dos vídeos interativos, mas representa um fator importante para a adaptação da navegação da rede. Uma boa construção de vídeo interativo é necessária para se alcançar bons resultados da adaptação no que diz respeito a seleção de um bloco dentre os existentes na hierarquia.

1.4.2 Avaliação e Confiabilidade

A quantização da rede está diretamente relacionada à avaliação e à confiabilidade dos nodos que a compõem. O cálculo para ambos os valores é feito em duas etapas: direta e indireta. A avaliação direta ocorre quando um estudante responde uma questão do nodo em que se encontra. Isto é, o estudante é avaliado pelo sistema por meio de sua resposta, essa avaliação direta é sucedida pela avaliação indireta do nodo, que corresponde as operações de fecho, e representa os processos de inferência sobre a rede para tentar prever estados futuros. O resultado A da avaliação do nodo é dado pela média ponderada das notas das avaliações direta (A_0) e indireta (A_{0M}), tendo como fator de ponderação a confiabilidade (C), como mostra a Equação 1.1.

$$A = CA_0 + (1 - C)A_{0M} \tag{1.1}$$

Para o cálculo da avaliação indireta, utiliza-se a nota da avaliação do nodo de origem A_1 e uma constante $k_a, (0 < k_a < 5)$, que representa o grau de liberdade para se alterar a nota A_0 do nodo, como apresentado na Equação 1.2.

$$A_{0M} = A_0 + (A_1 - A_0)\frac{k_a}{5} \tag{1.2}$$

O valor da avaliação direta é a média aritmética das notas das questões respondidas do nodo. Por exemplo, se um nodo da rede possui sete questões, o estudante respondeu apenas quatro e tirou nota seis em duas delas e oito nas outras duas, a nota da avaliação direta então será sete. As outras três questões que não foram respondidas não influenciam diretamente o cálculo da avaliação, mas servem para o cálculo da confiabilidade. A avaliação final calculada e finalmente dada pela Equação 1.3.

$$A_0 = CA_0 + (1 - C)\left[A_0 + (A_1 - A_0)\frac{k_a}{5}\right] \tag{1.3}$$

A confiabilidade C do nodo é calculada também por meio de uma média ponderada das duas etapas, direta e indireta, como mostra a Equação 1.4.

$$C = \frac{(n_a + 1)C_0 + C_{0M}}{n_a + 2}, \tag{1.4}$$

onde n_a é o número de questões respondidas pelo usuário, C_0 representa a confiabilidade anterior do nodo e C_{0M} é a confiabilidade da avaliação indireta proposta, que deve aumentar nos casos em que A_0 for um valor próximo de A e diminuir nos casos em que A_0 e A forem muito distantes.

A confiabilidade direta do nodo (ρ_{KR21}) é calculada por meio de uma inferência estatística, avaliando a correlação de todas as questões da avaliação, questão a questão. Isso é feito por meio da fórmula 21 de correlação de Kuder-Richardson, como mostra a Equação 1.5

$$\rho_{KR21} = \frac{k}{(k-1)}\left[1 - \frac{\mu(k-\mu)}{k\sigma^2}\right], \tag{1.5}$$

onde k é o número de questões, μ é a média dos escores obtidos pelos alunos e σ^2 é a variância corrigida em torno da média, dada pela Equação 1.6

$$\sigma^2 = \frac{\sum x^2}{n-1}, \tag{1.6}$$

onde x representa a diferença entre o escore do estudante e a média dos estudantes para aquela questão. Nesse sentido, é possível ver que as questões que não foram respondidas terão um valor x muito alto, o que influencia negativamente no resultado confiabilidade.

Da mesma forma, para o cálculo da confiabilidade da avaliação indireta, a distância entre os escores do estudante deve ser levada em consideração, já que, se houver uma variação muito grande entre duas avaliações, isso implica em uma confiabilidade baixa. Além disso, o próprio valor da confiabilidade do nodo de origem influencia o cálculo da nova confiabilidade, como é mostrado na Equação 1.7.

$$C_{0M} = C_1\left(1 - \frac{|A_0 - A_1|}{10}\right) \tag{1.7}$$

É importante ressaltar que a avaliação de um nodo varia entre 0 e 10, por isso a diferença entre as avaliações A_0 e A_1 deve ser dividida por 10 para que o valor final da confiabilidade indireta C_{0M} esteja sempre em um intervalo entre 0 e C_1.

Para o cálculo final da confiabilidade, tem-se a Equação 1.8.

$$C = \frac{(n_a+1)C_0 + C_1(1 - 0,1|A_0 - A_1|)}{n_a + 2} \tag{1.8}$$

1.4.3 Influência Temporal

Para que a rede se adeque aos acessos atuais dos nodos, é realizado um decréscimo percentual da avaliação de um nodo quando um aluno não o acessa durante determinado período de tempo. A ideia é reduzir o grau de interferência das avaliações passadas sobre uma nova retomada do curso, quando o aluno passa algum tempo sem acessá-lo. Esse

cálculo é feito de acordo com a Equação 1.9.

$$A = (1 - p_e)A_0 \tag{1.9}$$

onde p_e é o fator de decaimento da avaliação e deve estar contido no intervalo $[0, 1]$.

1.4.4 Quantização das Ligações entre Nodos

Como dito antes, a quantização da rede é gerada a partir da menor distância entre os nodos da rede. Esse cálculo é feito levando em consideração o campo tridimensional. Em um primeiro momento, todos os nodos da rede estão em ordem, em $z = 0$ e N_4 sendo o nodo atual. O exemplo considera três alterações de nível dos nodos da rede que se referem a blocos coesos, direção e avaliação.

Primeiramente o nodo N_4 é colocado em um nível superior da rede de modo que os outros nodos não ultrapassem o nodo atual após as modificações de suas posições (Fig. 4a). Posteriormente, todos os nodos que pertencem ao mesmo bloco coeso de N_4 sofrem uma elevação de nível (Fig. 4b).

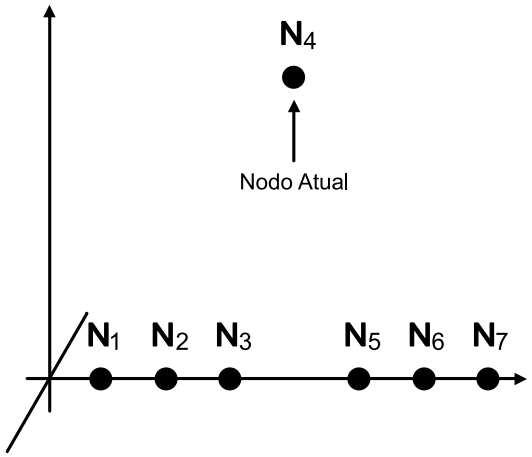
Considerando neste exemplo que o estudante tenha obtido um escore superior ou igual ao considerado satisfatório, todos os nodos posteriores ao atual sofrem uma elevação de nível (Fig. 4c), pois a direção da avaliação indica um avanço positivo. O próximo passo é reduzir proporcionalmente o nível dos nodos com base nas suas respectivas avaliações e confiabilidades (Fig. 4d).

Com base na nova configuração obtida, basta fazer o cálculo da menor distância entre o nodo atual e os demais para se obter a quantização da rede, segundo a fórmula da menor distância para pontos tridimensionais. Sendo assim, podemos fazer a análise também para a distância ao quadrado, como mostrado na Equação 1.10.

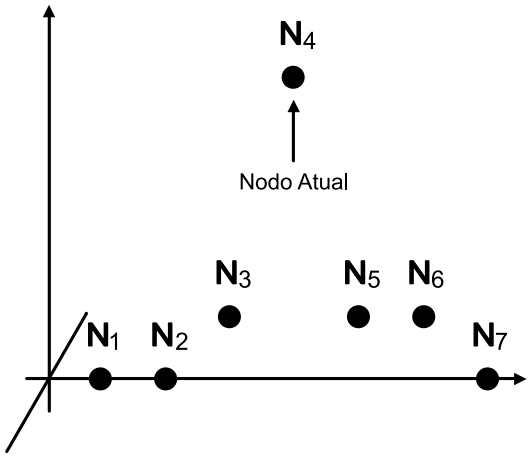
$$d^2 = (x_a - x_i)^2 + (y_a - y_i)^2 + (z_a - z_i)^2 \tag{1.10}$$

onde (x_a, y_a, z_a) é a posição do nodo de origem e (x_i, y_i, z_i) é a posição de algum outro nodo da rede que se pretende calcular a distância. Essa teoria pode ser utilizada para redes n-dimensionais sem prejuízo, pois d^2 é o quadrado da norma euclidiana do vetor distância entre dois nodos da rede (FRAGELLI, 2010).

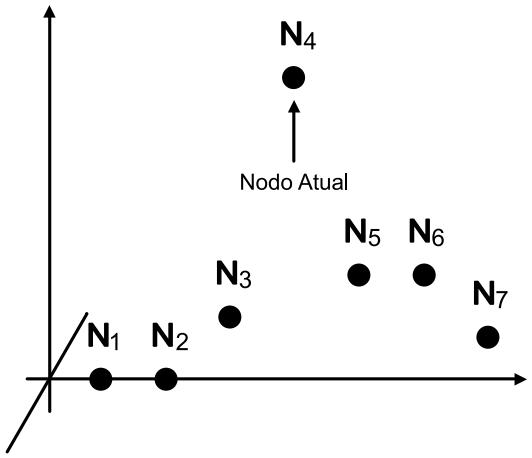
Para que aconteça a quantização da rede, é preciso definir os coeficientes de alteração do nível z e de posicionamento (x_a, y_a) para os nodos, com a finalidade de gerar uma modelagem flexível para diferentes modelos pedagógicos. Fragelli (2010) ratifica essa afirmação por meio da apresentação de duas propostas de redes quantizadas: behaviorista e cognitivista.



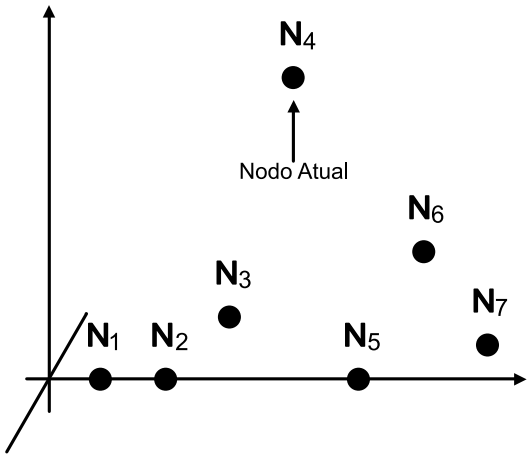
(a) Deslocamento do nodo atual.



(b) Deslocamento dos blocos coesos.



(c) Deslocamento segundo a direção.



(d) Deslocamento segunto as avaliações.

Figura 4 – Processo de quantização da rede.

1.4.5 Rede Behaviorista Quantizada

Para uma rede behaviorista, os nodos estão alinhados em x e são dados pelos valores definidos pelo professor ao compor seu curso. Desse modo, a quantização está relacionada à análise de z para o posicionamento dos nodos como apresentado na Equação 2.11.

$$(x, y, z) = (x_a, 0, \sum_{i=1}^{m} k_i) \tag{1.11}$$

Fragelli propõe a utilização de três coeficientes k_i para a quantização da rede:

- k_1 Blocos Coesos;
- k_2 Direção e Expertise;
- k_3 Avaliação e Confiabilidade.

O coeficiente k_1 avalia se um nodo faz parte do mesmo bloco coeso do nodo atual N_a e define um limite superior de modo que os nodos externos ao bloco coeso não fiquem impedidos de serem acessados.

$$k_1(N_i) = \begin{cases} p \sum_{j=1}^{k} N_j & \forall N_j \in S(N_a, N_i) \\ 0 & \text{c.c.} \end{cases} \tag{1.12}$$

Na Equação 1.12 $S(N_a, N_i)$ é o conjunto de blocos coesos que contém N_a e N_i, já que podem pertencer a mais de um bloco coeso e k_1 é o coeficiente do nodo N_i calculado por meio do somatório dos elementos de blocos coesos que contenham N_a e N_i. Para o escopo deste trabalho, o professor deverá se encarregar de gerar esse coeficiente por meio das ligações entre os nodos no momento da criação de um curso. Entretanto, este conceito é importante para se entender a possibilidade de adequação da rede para a inserção de um agente pedagógico que possa gerar esses coeficientes analisando a interação do usuário na rede.

O coeficiente p está associado ao objetivo do aluno ao acessar um curso, se ele fizer parte do público-alvo para qual o curso foi designado, este aluno poderá seguir o planejamento normal do curso, acessando ordenadamente os nodos.

O coeficiente k_2 é calculado segundo a avaliação do nodo atual N_a e o fluxo normal indicado pelo autor. Para um nodo N_i, o valor de k_2 é dado pela Equação 1.13.

$$k_2(N_i) = \begin{cases} v & \text{se } A_{N_a} \geq p_b \varepsilon_{N_a} \text{ e } x_{N_i} > x_{N_a} \\ v & \text{se } A_{N_a} < p_b \varepsilon_{N_a} \text{ e } x_{N_i} < x_{N_a} \\ 0 & \text{c.c.} \end{cases} \tag{1.13}$$

Para determinação do valor de v, deve-se verificar o valor necessário para provocar uma tendência de seguir o fluxo normal do curso. Para isso é necessário treinar a rede, semelhante ao processo de treinamento das redes neurais. O coeficiente p_b indica qual o percentual da expertise ε_{N_a} seria suficiente para uma avaliação satisfatória do nodo. É importante perceber que o coeficiente v depende da avaliação ser satisfatória e do nodo N_i anteceder ou proceder ao nodo N_a, pois uma avaliação não satisfatória deve indicar que nodos antecedentes devem ser acessados. Nessa mesma lógica, nodos posteriores devem ser indicados caso a avaliação do nodo N_a seja satisfatória.

O coeficiente k_3 tem o objetivo de diminuir a chance de um nodo com avaliação satisfatória e confiabilidade alta ser acessado. Sendo assim, este coeficiente é dependente dos valores de k_1 e k_2 calculados anteriormente. A solução proposta por Fragelli envolve a relação linear demostrada na Equação 1.14.

$$k_3(N_i) = -\min\left(\frac{\min(\frac{C_{N_j}}{C_0}, 1) A_{N_j}}{p_b \varepsilon_{N_j}}, 1\right)(k_1 + k_2) \tag{1.14}$$

Desse modo, se um nodo possui avaliação satisfatória e confiabilidade superior à confiabilidade mínima C_0, o valor de k_3 será $-(k_1+k_2)$ e o nodo retornará para $z = 0$. Em qualquer

outra situação, a redução será proporcional aos valores da avaliação e da confiabilidade do nodo N_i.

1.4.6 Rede Ausubeliana Quantizada

No caso de uma rede levando em consideração a teoria de Ausubel, são utilizados os mesmos coeficientes de nível da rede behaviorista, com uma modificação que é a utilização das posições dos nodos no mapa conceitual construído pelo professor. Assim, a distribuição espacial dos nodos no momento da construção do curso é bastante útil na quantização da rede que assumiria a forma da Equação 1.15.

$$(x, y, z) = (x_a, y_a, \sum_{i=1}^{m} k_i), \tag{1.15}$$

onde (x_a, y_a) representam a posição do nodo no mapa conceitual da rede. O grande avanço desta proposta é a possibilidade da modificação do mapa conceitual para cada estudante distinto que acessa a rede, suas características influenciam também a disposição dos elementos. Uma grande discussão sobre o uso dos mapas conceituais é que o mapa conceitual construído pelo professor nem sempre representa o mapa que seria feito pela estrutura cognitiva do estudante, por isso é importante o desenvolvimento de um sistema que permita a adaptação do mapa conceitual segundo as impressões do estudante.

Com base no aporte teórico educacional apresentado, é necessário compreender também os métodos e técnicas do estudo em engenharia de software que proporcionem meios suficientes para que o sistema proposto possa ser desenvolvido, sendo estes apresentados a seguir.

1.5 Arquitetura de software

Ao passo que os sistemas de software se desenvolvem em tamanho e complexidade, os problemas de projeto vão além dos algoritmos e estruturas de dados: projetar e especificar a organização geral do software surge como um novo problema. Falhas estruturais incluindo mecanismo global de controle; protocolos de comunicação, sincronização e acesso de dados; distribuição física; composição dos elementos do projeto; escalabilidade e performance; e seleção dentre as alternativas de design. Esse é o nível arquitetural do projeto de software (SHAW; GARLAN, 1996).

Este nível de projeto de software possui um grande conjunto de trabalho, decisões sobre padrões de projeto, utilização de paradigmas, *frameworks* e modelos arquiteturais que são de suma importância para redução de retrabalhos e aumento da qualidade e manutenabilidade do produto final (KRASNER; POPE, 1988).

O modelo arquitetural MVC (do inglês *Model - View - Controller*) é um estilo fundamentado no modelo de arquitetura em camadas, primeiramente idealizado por Reens-

kaug (1979) para um projeto específico chamado *Smalltalk*. Tendo como objetivo separar o modelo de negócio abordado no sistema (camada *Model*), da camada de interface com o usuário (*View*), permitindo que esse modelo possa ser exibido de inúmeras e diferentes formas sem ter conhecimento de quem o está exibindo (KRASNER; POPE, 1988). A Fig. 5 ilustra o funcionamento da arquitetura MVC e as interfaces entre camadas.

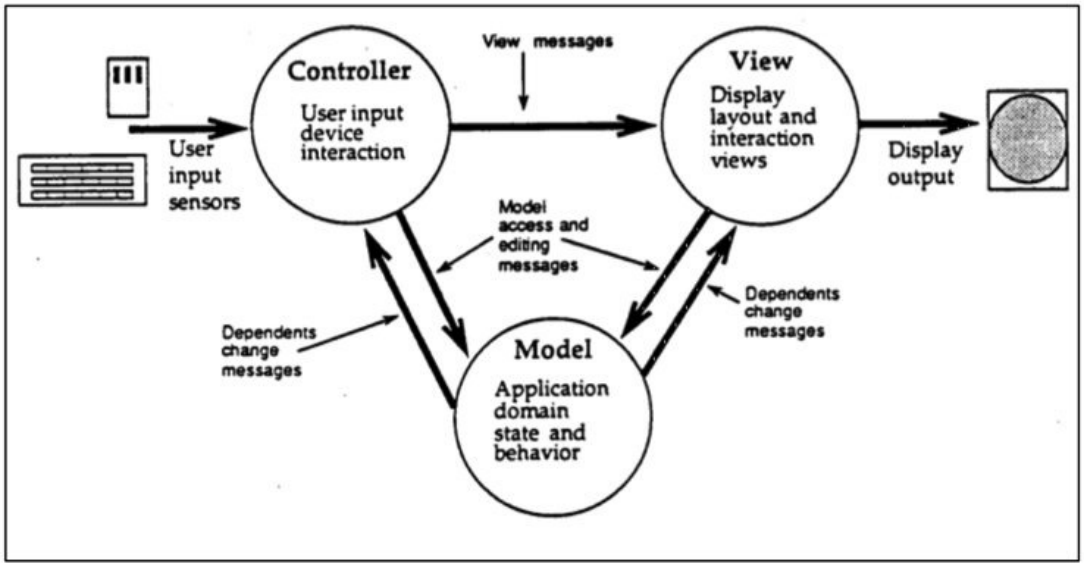


Figura 5 – Model - View - Controller.
Fonte: (KRASNER; POPE, 1988)

Para manter todas as *Views* e *Controllers* atualizadas segundo o estado da *Model*, foi definida também a noção de dependência. Os objetos do tipo *Controller* e *View* são registrados em uma lista de dependentes na *Model*, e quando ocorre alguma mudança no estado do modelo, estes são avisados através de troca de mensagens (KRASNER; POPE, 1988).

Anos depois, no final da década de 1980, o paradigma foi publicado como um padrão arquitetural para sistemas *web*, no sentido de que é interessante que o modelo não conheça a forma de apresentação, já que uma página pode ser exibida em diferentes dispositivos com diferentes resoluções e proporções de tela, e que deve se comportar de forma diferente sem comprometer o bom funcionamento do sistema (KRASNER; POPE, 1988), se tornando uma forte tendência para estas aplicações desde então.

Esta estrutura arquitetural foi utilizada inicialmente para renderização *server-side* (SHABAITAH, 2014), ou seja, o processo digital sobre a página é feito no servidor que provê o serviço *web*, que no início da década de 90, poucos computadores pessoais possuiam capacidade para executar o processamento das páginas depois de recebidas e que ainda hoje existem dispositivos com pouca capacidade de processamento que também fazem consultas a internet e dependem do servidor para renderização, como equipamentos de *e-books* e alguns *smartphones*(BAKER, 2006).

Com o desenvolvimento de navegadores mais robustos e com funcionalidades diversas, construir aplicações que dependem de renderização no cliente se tornou não somente factível, mas bastante popular (GOOGLE, 2015). Nesse sentido o estilo arquitetural MVC para aplicações *web* tem sofrido consideráveis mudanças, de modo que camadas da aplicação antes existentes apenas nos servidores passaram a executar papéis importantes também no cliente (GOOGLE, 2015).

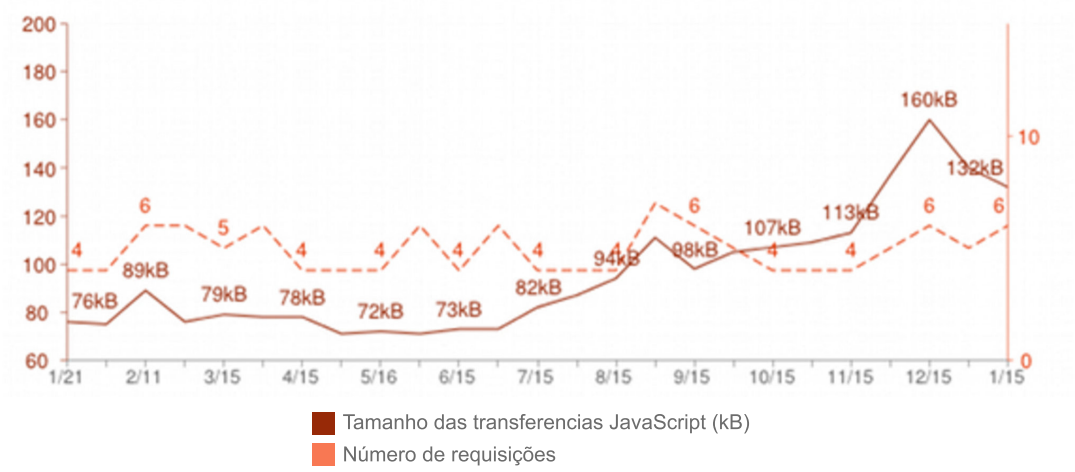


Figura 6 – transferências e requisições JavaScript.
Fonte: (GOOGLE, 2015)

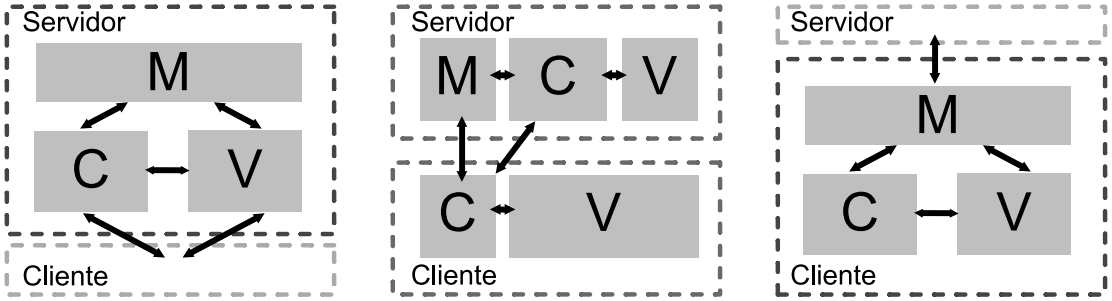
A Figura 6 mostra o crescimento da popularidade de arquivos *JavaScript* em páginas *web*, linguagem mais comumente usada para aplicações que são executadas no lado do cliente e que podem também gerenciar renderização.

Com base no que foi apresentado, é possível encontrar na literatura várias formas de implementar uma arquitetura MVC com renderização no servidor, no cliente ou em ambos. A figura 7 a seguir ilustra alguns exemplos genéricos encontrados em ferramentas bastante utilizadas atualmente (MEJIA, 2011; PETITIT; TRICOT, 2014; QUINN, 2015).

Uma arquitetura *web* tradicional, utiliza em geral renderização no servidor e retorna a página pronta para o cliente que interage por meio de controles providos pelo navegador, redirecionando as requisições ao servidor, que executa o processamento e retorna uma nova página ao cliete após a resposta à requisição ser construída. Para essa estrutura, a cada ação do usuário, uma nova requisição é criada e existe uma latência considerável entre executar a requisição e receber a resposta novamente no cliente (MEJIA, 2011).

Para uma arquitetura no cliente ou uma arquitetura mista, existem *frameworks* que provêm recursos suficientes para persistencia e controle no cliente. No caso da arquitetura mista, podem existir APIs no servidor implementados em diversas linguagens para gerenciar requisições e envios de dados ao cliente, além da possibiliade de responderem uma página renderizada (PETITIT; TRICOT, 2014; QUINN, 2015). Já para aplicações

que possuem um MVC completo no cliente, o servidor pode servir meramente como um banco de dados, que provê as informações necessárias ao cliente por meio de uma API *publish-subscribe*[1] (QUINN, 2015).



(a) *server-side.* (b) misto. (c) *client-side.*

Figura 7 – Estilos Aquiteturais MVC.

Apesar de suas diferenças e semelhanças, cada modelo possui vantagens e desvantagens. No caso de renderização no servidor, uma grande vantagem é a independência do cliente, que caso possua pouca capacidade de processamento, conseguirá executar a aplicação sem grandes problemas. Já para aplicações com renderização no cliente, é necessário que o dispositivo seja veloz o suficiente para executar a aplicação. Apesar disso, aplicações com renderização no cliente possuem geralmente uma interface mais rica e usável (UBL, 2015). No caso deste trabalho, é proposta uma arquitetura mista, detalhada no Capítulo 2 sobre o desenvolvimento.

Independente de onde aconteça a renderização, o MVC possui vantagens como possibilidade de reuso de código, separação entre níveis conceituais e menor acoplamento entre as camadas, possibilitando maior manutenabilidade e qualidade do código produzido (KRASNER; POPE, 1988). Entretanto, a qualidade do *software* não é medida unicamente por sua arquitetura, testes são meios menos abstratos de se garantir a qualidade do software desenvolvido. A seguir, descorre-se sobre estratégias e tipos de testes encontrados na literatura.

1.6 Testes de Software

O processo de teste de software procura determinar se um sistema atinge suas especificações e funciona corretamente no ambiente para o qual foi projetado, tendo como objetivo identificar falhas durante o desenvolvimento para que possam ser corrigidas antes da entrega do produto final (NETO, 2005).

Para entender o conceito de testes, é necessário compreender a diferença entre defeito, erro e falha. Um defeito se encontra no meio físico é uma instrução de um passo,

[1] Padrão utilizado em sistemas de mensageria, como SMS ou MQTT. Permite que um cliente registrado em um canal receba notificações quando um outro cliente faz publicações no mesmo canal.

ou deifinição de dados incorreta. Um erro está no âmbito da informação, na construção de um software diferente do que foi especificado, sendo qualquer passo intermediário diferente do que é esperado. Uma falha, por sua vez, é o comportamento inesperado que afeta diretamente o usuário final da aplicação, podendo inviabilizar o uso do sistema (COMMITTEE et al., 1990), como ilustra a Figura 8.

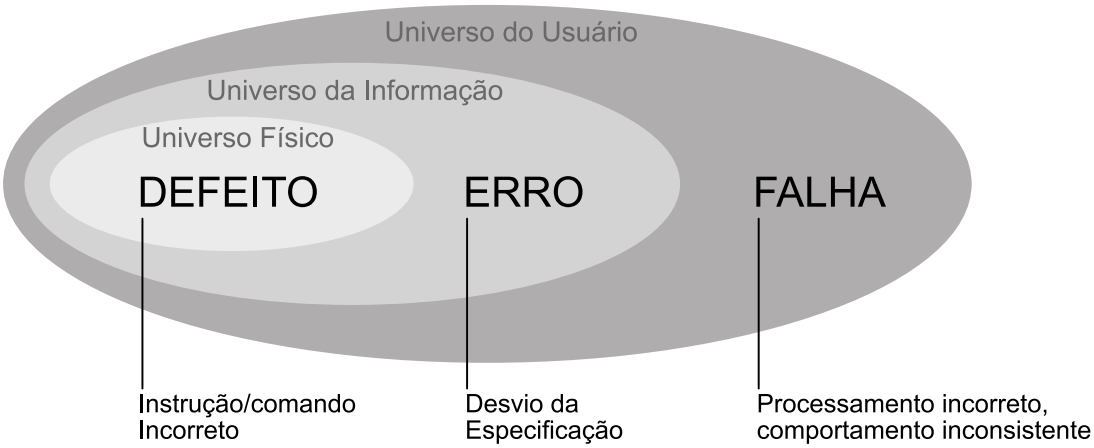


Figura 8 – Defeito × Erro × Falha.

Testes podem ser projetados de dois pontos de vista: estrutural ou funcional. Os testes funcionais ignoram o mecanismo interno de execução e se concentram nos resultados gerados com o propósito de encontrar não conformidades com a especificação da funcionalidade. Testes funcionais são direcionados ao usuário e devem se preocupar unicamente com a funcionalidade. Já os testes estruturais analisam o funcionamento interno em diferentes níveis, podendo envolver todo o sistema, ou apenas partes dele. Para Naik e Tripathy (2008), existem 3 níveis de testes estruturais, sendo eles de unidade, integração e sistema.

Testes unitários são utilizados para testar de forma isolada funções, métodos ou classes. Após garantir que as unidades funcionam de forma satisfatória, os módulos são agrupados para se construir sub-sistemas maiores que são testados por meio das técnicas de teste de integração. Testes unitários são importantes pois, uma vez que o código está integrado a outros módulos, encontrar a causa de uma falha se torna mais difícil (NAIK; TRIPATHY, 2008). Existem diferentes ferramentas de teste unitário para diversas linguagens, porém a maioria provê recursos semelhantes para construção de acertivas a fim de verificar se os resultados retornados pelo código testado estão de acordo com a especificação proposta. Algumas dessas ferramentas são JUnit (JUNIT, 2015), ActiveSupport (RUBY ON RAILS, 2015a), Mocha (MOCHA WEB, 2015).

As ferramentas citadas acima podem ser utilizadas também para construção de testes de integração. Testes estes que são efetuados de modo incremental, adicionando e testando, a cada ciclo, um novo módulo do sistema. Esse processo é repetido até que todos os módulos tenham sido integrados e testados. Esse nível de teste examina explicitamente

as interfaces entre unidades a fim de verificar se o sistema é capaz de suportar testes do nível de sistema (NAIK; TRIPATHY, 2008).

Os testes a nível de sistema englobam testes básicos de instalação e configuração; testes funcionais estensivos sobre todo o sistema; testes de robustez para verificar quão bem o sistema se recupera de situações de falha; testes de interoperabilidade, performance e escalabilidade e também testes de stress, para verificar os limites do sistema e em quais situações a falha ocorre. Muitos dos testes de sistema dependem diretamente do ambiente no qual o software será instalado e utilizado, pois necessitam aproximar os resultados para um ambiente o mais próximo do real possível (NAIK; TRIPATHY, 2008).

1.7 Metodologia *Scrum*

O *Scrum* é um *framework* ágil para gerenciamento de projetos, fundamentado no empirismo. O empirismo afirma que o conhecimento provém da experiência e de tomadas de decisões baseadas no que é conhecido. Desse modo, o *Scrum* utiliza do método da inspeção e adaptação para reagir a mudanças e melhor lidar com a complexidade e riscos do projeto. Outro aspecto importante do método é o foco no aprendizado e difusão do conhecimento, em que todos os integrantes do time cooperam para o aprendizado coletivo e para um melhor desenvolvimento do produto (SCHWABER; SUTHERLAND, 2013).

O *Scrum* pode ser utilizado em diferentes contextos. Entretanto, surgiu para suprir uma demanda clara do processo produtivo da indústria de software, na qual inúmeros projetos falharam pela ineficiência dos processos produtivos em cascata, mais utilizados até meados dos anos 90, em projetos que custaram milhões e resultaram em fracasso para grandes organizações como *FBI (Federal Bureau of Investigation)* ou *NPR (National Public Radio)* (SUTHERLAND, 2014).

Para este trabalho, foram levados em consideração quatro princípios básicos da metodologia listados por Sutherland (2014), sendo eles:

1. **Planejar é útil. Seguir cegamente os planos é burrice.** Simplifique o processo de planejamento e aja o quanto antes. Reavalie o seu planejamento com frequência, especular sobre o desconhecido é pouco eficaz, devido à dificuldade de se controlar inúmeras variáveis do processo.

2. **Inspeção e Adaptação.** De tempos em tempos é necessário rever o trabalho que foi desenvolvido e verificar se o caminho traçado ainda está correto, ou se existem formas melhores de fazê-lo.

3. **Mudar ou Morrer.** Ficar preso em métodos antigos de mandar e controlar ou manter uma previsibilidade rígida resultará no fracasso. A mudança gera vantagens estratégicas e comerciais e deve sempre ser bem-vinda.

4. **Fracasse rápido para corrigir o problema o quanto antes.** Todo o trabalho que gera um desperdício de esforço deve ser eliminado, possibilitando que produto real seja entregue ao usuário com o tempo e esforço economizados.

Claramente, os termos utilizados pelo autor são metafóricos, mas indicam de forma simples a ideia por trás dos tópicos: A possibilidade de mudança e adaptação do *framework* para diferentes times e organizações, possibilitando a adequação do método para a sua utilização.

Aplicado a este trabalho, o *Scrum* sofreu várias adaptações para que pudesse ser utilizado. Dos papéis definidos no *Framework*, apenas o dono do produto e o time de desenvolvimento foram mantidos, sendo desempenhados pelo orientador e pelo autor, respectivamente. Com relação às práticas sugeridas por Sutherland (2014), apenas a *sprint review* e o planejamento da *sprint* foram mantidas, com a participação do dono do produto e do time de desenvolvimento. Para a organização das histórias a serem cumpridas a cada *sprint* foi utilizado o *kanban* para controlar o fluxo de trabalho e estimar as próximas interações.

Por fim, uma última prática que necessita de destaque para qualquer desenvolvimento ágil de software, é o processo de integração contínua. Os times que utilizam essa prática visam alcançar dois resultados primários: a redução do custo para se requerido para cada episódio de integração dos componentes do sistema e a possibilidade de entrega de produto funcional a qualquer momento do desenvolvimento (AGILE ALLIANCE, 2015).

Na prática, estes objetivos requerem um processo de integração que seja passível de reprodução, pelo menos, em sua grande maioria, e que seja "automático". Isto é conseguido através de ferramentas de controle de versão, políticas e convenções da equipe e ferramentas projetadas especificamente para ajudar a alcançar a integração contínua (AGILE ALLIANCE, 2015).

2 Desenvolvimento

Este capítulo engloba questões do desenvolvimento do produto, sendos estas: modelagem do domínio, suporte tecnológico, arquitetura do software, testes, gerência de configuração e aspectos gerais.

2.1 Modelagem do Domínio

O modelo do domínio do software pretende apresentar os elementos que compõem a lógica do sistema e como eles foram organizados para que os objetivos do trabalho pudessem ser alcançados. É importante ressaltar que estes modelos apresentados foram evoluídos no decorrer do desenvolvimento até alcançarem um nível estável, já que o método seguido foi interativo e incremental segundo as práticas do *Scrum*.

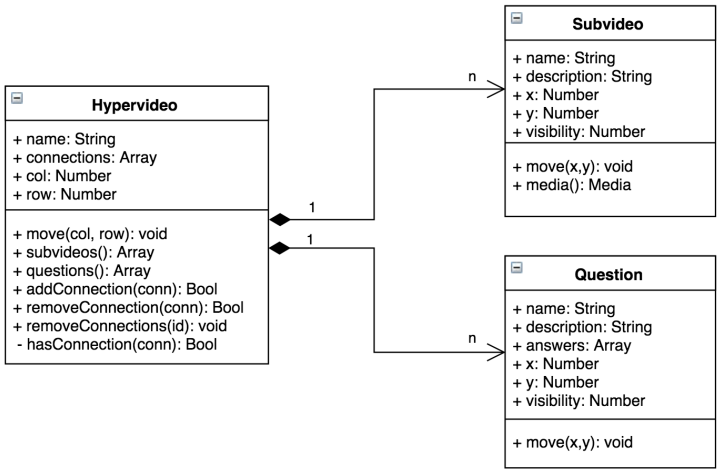


Figura 9 – Diagrama do vídeo interativo proposto.

Primeiramente, foi modelado um vídeo interativo que pudesse ser utilizado na construção de um curso. Levando em consideração os princípios do design multimídia, os vídeos interativos possuem pouco espaço para textos escritos na tela; são formados por subvídeos interligados conforme definido pelo professor autor; possuem questões a serem respondidas ao longo da apresentação do vídeo e devem ser adaptativos de acordo com os diferentes perfis de usuário. Além disso, um vídeo interativo deve possuir as informações necessárias para que ocorra a quantização da rede, ou seja, para cada usuário que o acesse, deve existir uma avaliação e uma confiabilidade atribuidos a ele, bem como uma posição em um plano bidimensional referente ao mapa conceitual do curso ao qual o vídeo interativo pertence.

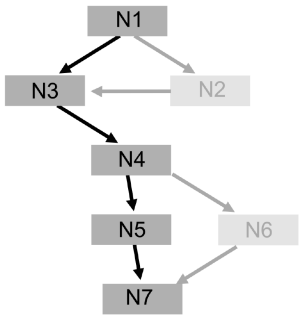


Figura 10 – Adaptação do hypervideo por meio da ocultação dos nodos N2 e N6.

Cada subvídeo possui informações básicas sobre a mídia de vídeo que será apresentada por ele, como título e uma breve descrição do conteúdo. Além disso, possui um nível de visibilidade para ser apresentado apenas ao perfil específico para o qual foi projetado, garantindo assim a capacidade de adaptação a nível de conteúdo. A Fig. 9 apresenta um diagrama do vídeo interativo, que no sistema é denominado Hypervideo.

Por meio da Figura 9, é possível perceber que existe uma lista de conexões para cada Hypervideo. Essa lista representa o conjunto de arestas do grafo que tem como vértices os subvídeos e questões pertencentes a o vídeo interativo. Como visto no capítulo anterior, as teorias behavioristas apresentam o conteúdo de forma sequencial e linear, já as linhas cognitivistas compreendem estruturas de decisão e caminhos distintos para cada aprendiz. A Figura 10 mostra como essa estrutura de grafo pode se adaptar para adequar-se a ambas as formas de apresentação.

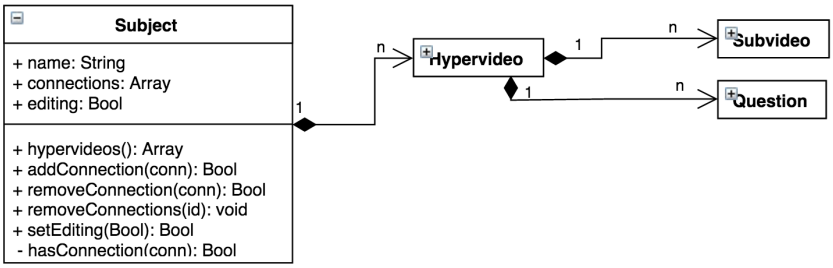


Figura 11 – Diagrama da estrutura do curso proposto.

Para que fosse possível desenvolver uma adaptação da navegação que utilizasse a QRN, foi necessário modelar também a estrutura do curso que comportasse os hypervídeos e as informaçõs necessárias para os cálculos. É proposto que um curso possua também uma estrutura de grafo que tenha como vértices os seus Hypervideos, e que as conexões criadas pelo professor representem as ligações conceituais entre os tópicos apresentados em cada hypervídeo. A Figura 11 apresenta o modelo relativo ao curso que foi denominado *Subject*.

A próxima modelagem feita foi referente aos dados do usuário: cursos que assiste, cursos construídos por ele, percentual de conclusão de um curso, questões respondidas,

hypervideos assistidos, etc. Todas essas informações requereram modelos que vinculassem cada elemento de um curso ao usuário para que se pudesse calcular os coeficientes de quantização para os nodos da rede. A Figura 12 mostra o modelo de domínio completo do *software*.

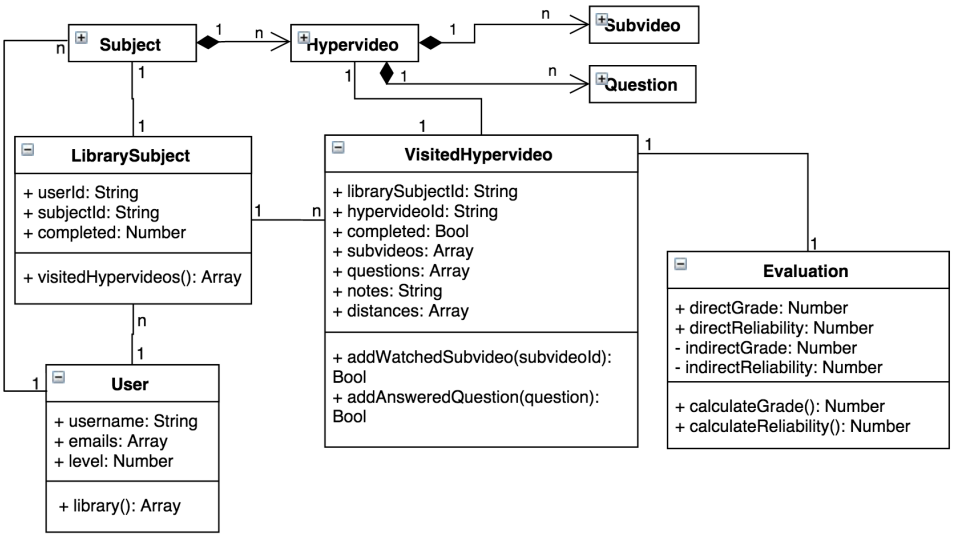


Figura 12 – Diagrama sobre informações referentes a relação usuário × curso.

Como a quantização da rede é definida de forma diferente para cada usuário, os cálculos da avaliação e confiabilidade diretas e indiretas devem estar vinculados ao objeto que interconecta o usuário ao nodo da rede a qual se pretende calcular a quantização. Dessa forma, para cada usuário que se cadastre na rede e assista a algum curso, uma quantização diferente será gerada segundo a sua interação com o sistema.

2.2 Suporte Tecnológico

Sobre o ponto de vista arquitetural, foram elicitadas ferramentas que proporcionassem facilidades para o desenvolvimento do sistema. Foram selecionados três *frameworks* que possibilitam o desenvolvimento de aplicações *web* com certo nível de abstração arquitetural, que são mantidos segundo licenças de software livre e que possuem comunidades ativas. Sendo estes: Grails (GRAILS ORG., 2015), Ruby on Rails (RUBY ON RAILS, 2015b), e Meteor (METEOR ORG., 2015a).

Para avaliar as ferramentas selecionadas, foi construído um cenário de prova de conceito, com a finalidade de verificar o esforço necessário para se implementar um pequeno software de manutenção de usuários. Os resultados coletados são descritos na tabela 1 abaixo:

Os *frameworks* Grails e Ruby on Rails possuem estilos arquiteturais bastante semelhantes, já que ambos são orientados a convenção, e possuem ferramentas de terminal

Tabela 1 – Tempo para implementação da funcionalidade.

Framework	Tempo
Meteor	37 min
Rails	44 min
Grails	74 min

interativo para criação e adição de elementos do sistema (como modelos, visões, controladoras e *plugins*). Entretanto, o tempo para se desenvolver a funcionalidade com Ruby on Rails foi menor devido à configuração do ambiente de desenvolvimento e dependências do *framework*.

Uma das vantagens da utilização do Ruby on Rails para o desenvolvimento é a existência de uma comunidade bastante ativa e de documentação abrangente sobre vários dos plugins que são disponibilizados para a plataforma. Porém, a configuração de uma ferramenta para renderização no cliente não é trivial: as complicações com a construção de APIs para comunicação remota e com a utilização de plugins JSON para recuperar dados do servidor impossibilitaram a construção da funcionalidade com este recurso.

Para a plataforma Grails, os mesmos problemas foram encontrados para a construção de uma aplicação com renderização no cliente, além dos problemas com configuração de ambiente e tempo de desenvolvimento da funcionalidade. A vantagem em optar uma ferramenta Java é a facilidade de se adicionar recursos multiagentes para tutoria inteligente, já que existem ferramentas consolidadas como o JADE (JADE TEAM, 2015) que são mantidas nessa linguagem. Como o foco deste trabalho não abrange o desenvolvimento de agentes, este *framework* foi descartado.

O processo de instalação e utilização da plataforma Meteor foi o mais simples dentre as opções. Todas as dependências do sistema são mantidas por meio do gerenciador de pacotes do Node.js (NODE FOUNDATION, 2015) e foram instaladas sem nenhum empecilho.Além disso, o tempo para se implementar a funcionalidade proposta foi o menor, apresentou maior facilidade na construção de aplicações com renderização no cliente, possui comunidade ativa e mantém também mecanismos de teste e depuração para as aplicações, em sua maioria sob licenças de software livre compatíveis com GPL v3 adotada neste projeto.

A opção por uma ferramenta que facilite a renderização no cliente é uma necessidade da proposta para construção do curso, já que para criar um grafo de hypervideos em um curso, ou um grafo de subvideos e questões dentro de um hypervideo, é necessário construir uma biblioteca para permitir a utilização do sistema sem que, a cada interação, se gere uma nova renderização da página proveniente do servidor, e que evite perda de dados caso ocorra falha de conexão com a *internet*. Arquiteturas construídas com Ruby on Rails ou Grails não oferecem suporte a esse tipo de interação, exigindo bibliotecas

externas à plataforma e o conhecimento em linguagens de programação diferentes das utilizadas nativamente no *framework*, como *JavaScript* ou *CoffeScript*.

Com esta análise, e pela necessidade de domínio em apenas uma linguagem, o *JavaScript*, optou-se pela plataforma *Meteor*. Também pela possibilidade de incluir bibliotecas para componentes visuais *web*, como o *Polymer* (POLYMER AUTHORS, 2015), o que contribui para a construção de interfaces de usuário mais ricas. A lista completa de pacotes e ferramentas utilizadas se encontra no Apêndice A.

2.3 Arquitetura do Sistema

Antes de compreender a arquitetura do sistema proposto, é preciso compreender como o *framework* funciona e quais as possibilidades para expansão ou modificação da arquitetura proposta pela plataforma. Cada instância da aplicação no cliente registra serviços segundo padrão *publish-subscribe* para coleções providas pelo servidor (CASCIARO, 2014). Nativamente, o *framework* dá suporte apenas para registros de *publish-subscribe* para dados definidos em coleções no banco de dados, porém existem *plugins* que permitem composições entre diferentes coleções para um mesmo tópico de *publish*.

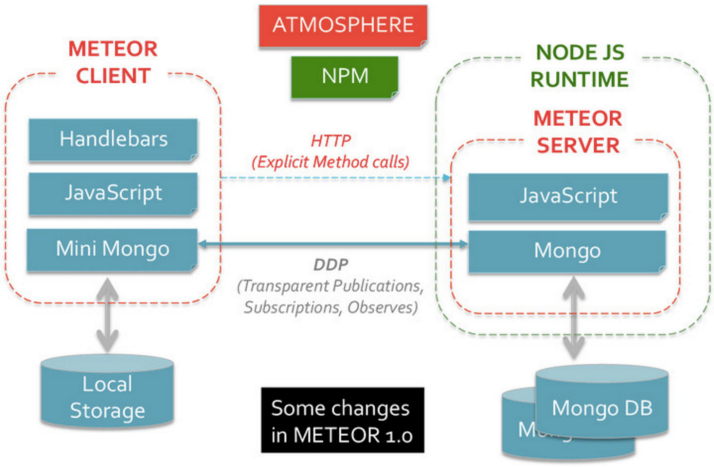


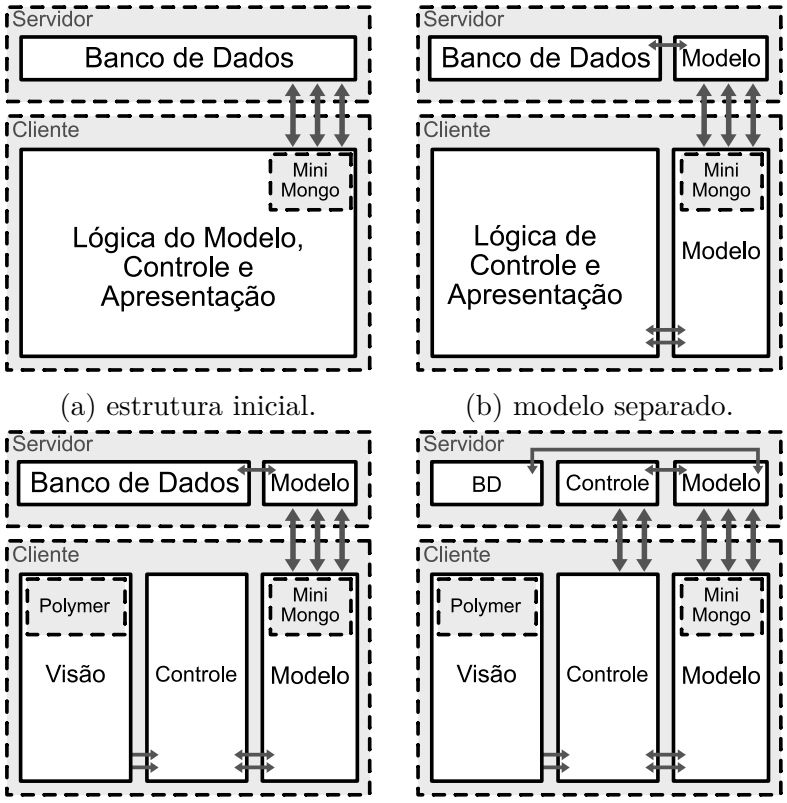
Figura 13 – Arquitetura Interna do *Framework Meteor*.
Fonte: (MONGODB, 2015)

O banco de dados utilizado pelo *Meteor* é o MongoDB, que dá suporte a arquivos JSON, é facilmente escalável e possui boa performance. A aplicação no servidor roda em um *container* Node.js e o *framework* gerencia as trocas de dados cliente-servidor por meio de uma implementação em JavaScript da API do MongoDB que roda no navegador, chamada MiniMongo (MONGODB, 2015). A Figura 13 mostra como o *Meteor* gerencia os dados da aplicação e como são providos os serviços para os clientes.

A plataforma não restringe, nem define um modelo arquitetural específico, abrindo espaço para que o desenvolvedor se encarregue de projetar de modo geral a arquitetura

da aplicação. Inicialmente, uma aplicação possui arquitetura semelhante a mostrada na Figura 14a, de forma que não existe separação nenhuma entre camadas lógicas, apenas entre servidor e cliente. A primeira forma da arquitetura proposta para o Sistema de Vídeos Interativos incluiu à aplicação uma camada de modelo, separando os dados do domínio, como ilustra a Figura 14b.

A lógica no cliente de uma aplicação *Meteor* é gerenciada por uma biblioteca de renderização nativa (i.e. *Blaze*), que responde reativamente às alterações que ocorrem nos dados locais. Essa característica permite a criação de aplicações nativamente reativas e gera requisições menores por utilizar um protocolo próprio e mais leve, denominado DDP (*Distributed Data Protocol*) (METEOR ORG., 2015b).



(a) estrutura inicial. (b) modelo separado.

(c) visão separada. (d) controle organizado.

Figura 14 – Evolução da arquitetura da aplicação

Apesar das vantagens, o *Blaze* possui alto acoplamento entre o controle e a visão, pois gerencia ambas as abstrações lógicas em um único arquivo, dificultando futuras manutenções da aplicação. Com a finalidade de se separar a lógica de controle e apresentação, foi adicionado ao sistema mais uma ferramenta, o *Polymer* (POLYMER AUTHORS, 2015), que utiliza dados do modelo recebidos pelo *Blaze* sempre que há alterações, e controla unicamente funções de apresentação, como posicionamento, cores, animações e outras operações da camada de visão. Esta versão é ilustrada pela Figura 14c.

Originalmente, aplicações *Meteor* sincronizam os dados do cliente com o servidor

automaticamente, sem que haja uma assinatura explícita dos dados (*i.e. subscription*). Com o crescimento da base de dados do servidor, se torna inseguro e inviável que os dados sejam compartilhados entre todos os clientes ativos da aplicação. Para corrigir esta falha, a camada de controle foi extendida ao servidor com a função de limitar os dados que eram enviados para cada cliente por meio do padrão *publish-subscribe*, sendo que apenas os dados necessários para a atual rota do usuário eram enviados para o cliente. Além disso, restrições de permissão para alteração dos dados foram criadas para manter a confiabilidade do banco. Esta é a versão atual da arquitetura, como ilustra a Figura 14d.

Agora que a arquitetura lógica foi descrita, deve-se verificar como esta foi implementada do ponto de vista de pacotes, com a finalidade de identificar a estrutura de diretórios e organização dos arquivos e dependências do projeto. O diagrama da Fig. 15 apresenta a organização dos pacotes.

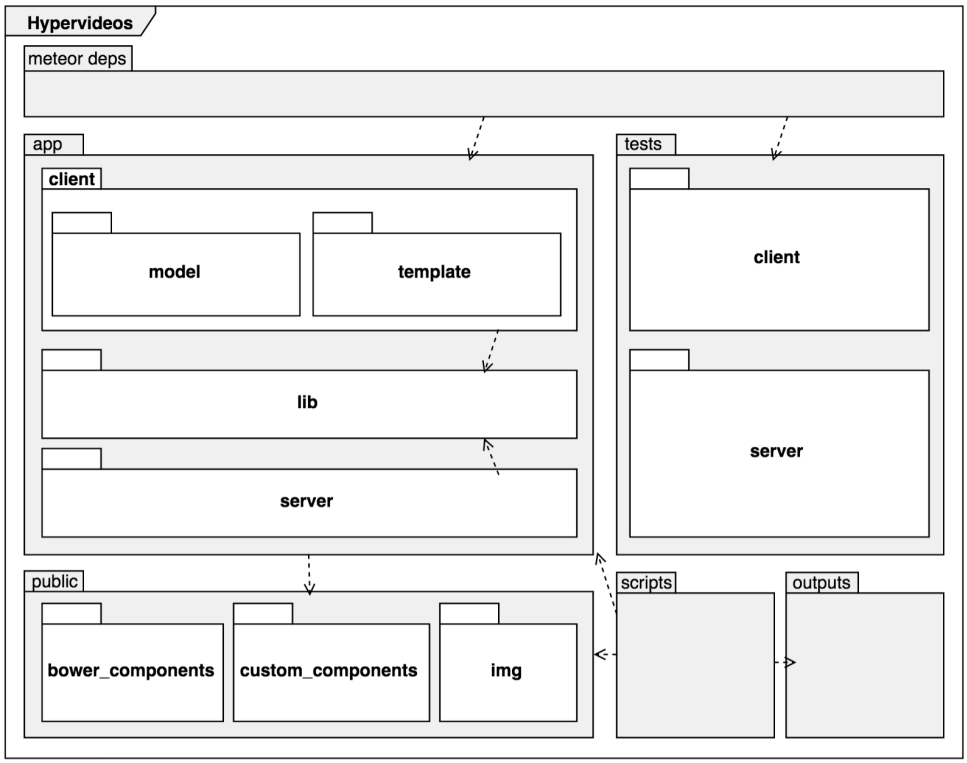


Figura 15 – Diagrama de pacotes da arquitetura.

É possível perceber que no diagrama não existe nenhum pacote denominado controle, ou visão. Isto se deve ao fato de que o diretório em que se encontram os arquivos do *Blaze* é geralmente denominado *template*. No caso desta aplicação esse pacote é a camada de controle do cliente. A camada de visão se encontra no pacote *public* juntamente com quaisquer outros recursos utilizados pela aplicação, como imagens.

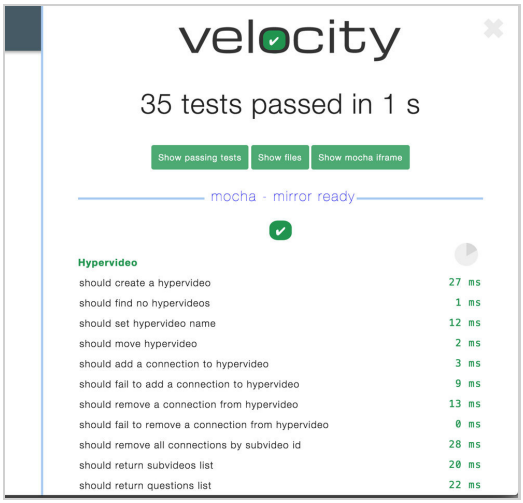
Além disso, não existe nenhuma separação em pacotes no servidor, para modelo e controle. Até o momento deste trabalho, para as operações do controle no servidor são necessários apenas dois arquivos e para o modelo, apenas um. Por esta razão, ainda não

houve a necessidade de se criar pacotes diferentes.

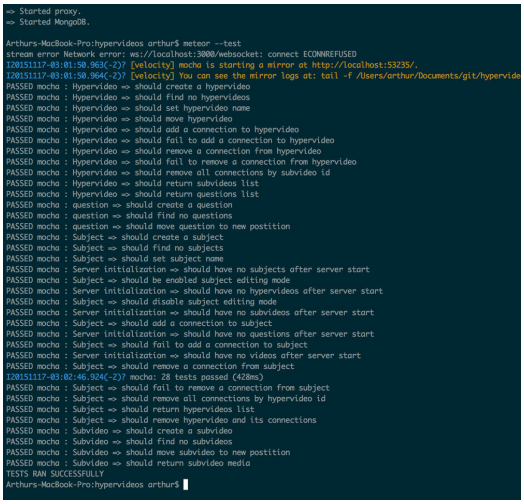
O pacote de *scripts* merece certo destaque pelo fato de: gerenciar testes dos componentes visuais; rodar a aplicação em modo de *profile* para análise de desempenho; e para gerar relatórios de cobertura e de análise estática de código, como aderência ao padrão de estilo *JavaScript* para o *Meteor*. Todas as saídas dos *scripts* são geradas no diretório oculto de *outputs* e servem como base para parte do processo de integração contínua, melhor explicado no tópico sobre gerência de configuração.

2.4 Testes

O *Velocity* é o *framework* oficial de testes para aplicações *Meteor*, suportando diferentes motores de teste — como Mocha (MOCHA WEB, 2015), Jasmine (XOLVIO, 2015b) ou Cucumber (XOLVIO, 2015a) — e permitindo que os resultados sejam exibidos em um componente reativo na interface da própria aplicação (Fig. 16a), ou via terminal (Fig. 16b), como um mecanismo para suporte em ambientes de integração contínua.



(a) componente *Velocity* html.



(b) *Velocity* via terminal.

Figura 16 – Visualização dos resultados de teste da aplicação.

Para construção dos testes da aplicação foi utilizado o motor *Mocha*, que fornece meios para a implementação de testes unitários e de integração que rodam com apenas um espelhamento da aplicação, reduzindo o consumo de memória se comparado com o Jasmine, que utiliza dois espelhos diferentes: um para testes unitários e outro para testes de integração (XOLVIO, 2015b). Essa característica é importante para um ambiente de integração contínua, pois reduz as configurações mínimas para a máquina que hospeda o serviço.

Além do *Velocity*, foi necessário configurar um mecanismo de testes para os componentes *web* construídos. A comunidade que mantém o *Polymer* mantém também uma ferramenta de testes unitários e um *plugin* para gerar relatórios de cobertura. O *script* para os testes dos componentes é encontrado no diretório de *scripts* e os relatórios de testes, como mostrado na Figura 17, são gerados dentro da pasta de *outputs*.

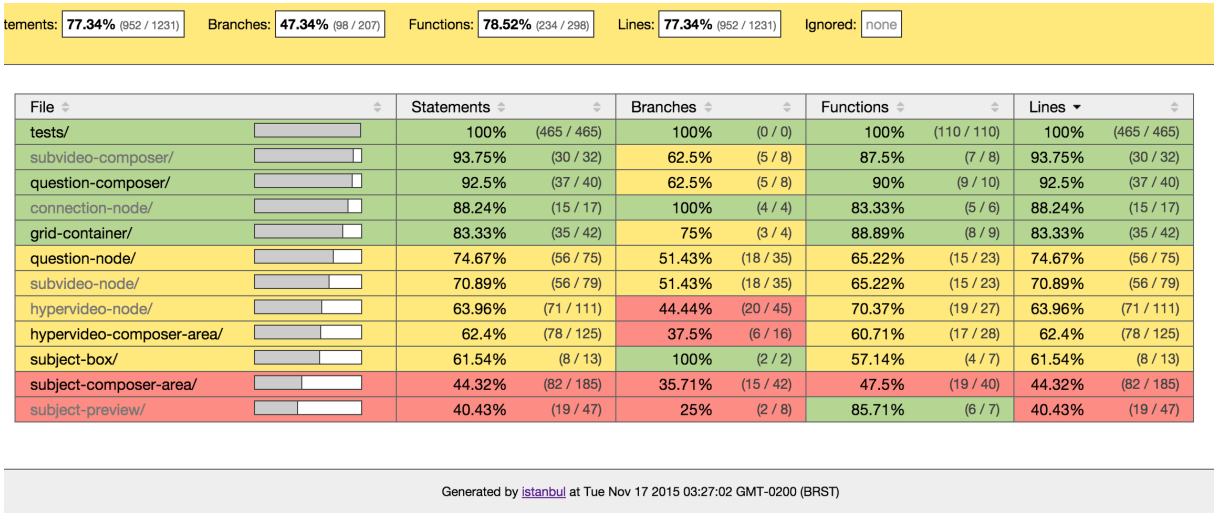


Figura 17 – Cobertura para componentes *Polymer*.

Após atualização para versão 1.2 do *Meteor*, devido à refatorações na lógica de criação de aplicações espelhadas, não existem *plugins* para gerar cobertura de código para a aplicação que seja compatível com o *Velocity*. Por essa razão, não são gerados relatórios de cobertura para a aplicação como um todo.

Além dos testes a níveis unitário e de integração utilizados, foi implementado um *script* de configuração de uma ferramenta para monitoramento de performance, como mostrado na Figura 18. A ferramenta foi mais utilizada para analisar o tempo de resposta para as operações de *publish* processadas no servidor, devido ao uso de consultas recursivas ao banco para composição de múltiplas coleções em apenas um tópico de *publish* (KADIRA ORG., 2015).

2.5 Gerência de Configuração

O controle de versão é a espinha dorsal de qualquer gerência de configuração, apoiando atividades de controle de mudanças e de integração contínua. Ferramentas de controle de versão atuam na identificação e armazenamento dos itens de configuração, e mantém versões durante todo o ciclo de vida do software, criando rótulos e ramificações no projeto (IEEE, 2014).

Existem diversas ferramentas que proporcionam diferentes formas de versionamento. Neste projeto, foi utilizado um repositório *git*, que mantém ramificações por meio

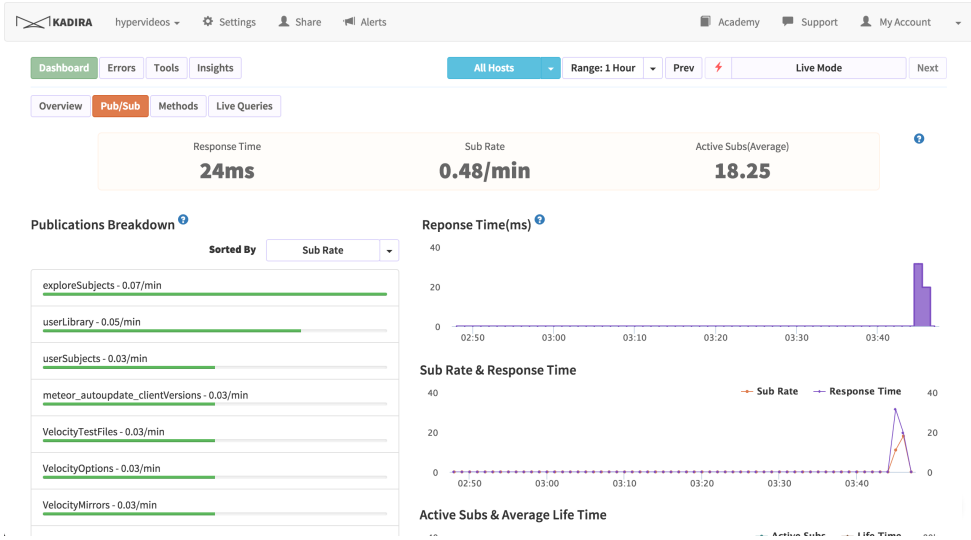


Figura 18 – Ferramenta para monitoramento de recursos utilizados pela aplicação.

de *branches* e registra mudanças na forma de *commits*. O repositório está mantido em uma ferramenta *on-line* chamada *Github*, que possui facilidades para gerenciamento e correção de falhas ou melhorias por meio de *issues* e possibilita a comunicação com ferramentas externas para gerenciamento de *builds*, por meio de *webhooks* (GITHUB, INC., 2015).

O processo de integração contínua tem como objetivo garantir que as mudanças no sistema sejam construídas, testadas e relatadas o quanto antes, para que falhas possam ser mais rapidamente encontradas e corrigidas. Para que o mecanismo funcione adequadamente, é preciso manter boas práticas entre a equipe de desenvolvimento e uma política de *branches*. Existem ferramentas que automatizam o processo de integração e possuem diversas configurações para análise de qualidade de código, cobertura de testes e diversos indicadores sobre o produto de software produzido. Neste projeto foi utilizado o *Jenkins* para configuração do ambiente de integração contínua da aplicação (JENKINS CI, 2015).

Para o repositório de código do projeto, foram criadas apenas duas *branches*, uma de desenvolvimento (*devel*), e a principal (*master*). Atualizações do desenvolvimento são sempre submetidas para a *branch devel*, e o ambiente de integração contínua se encarrega de gerar uma nova *build*, rodar os testes da aplicação, gerar os relatórios de análise estática de código e por fim, caso o resultado seja considerado satisfatório, as alterações são submetidas para a *branch master*.

No momento da construção do projeto o relatório de análise do código gerado no pacote de *scripts* foi carregado pelo *Jenkins* para popular o gráfico de alertas por *build*, proporcionando assim um histórico sobre a qualidade do código produzido e a sua evolução no decorrer do projeto.

Caso existissem falhas nas alterações que violassem as acertivas definidas para o conjunto de testes da aplicação, a *build* falharia, permitindo assim que as correções neces-

sárias pudessem ser feitas antes da entrega do produto. A Figura 19 mostra o ambiente de integração contínua configurado para o projeto.

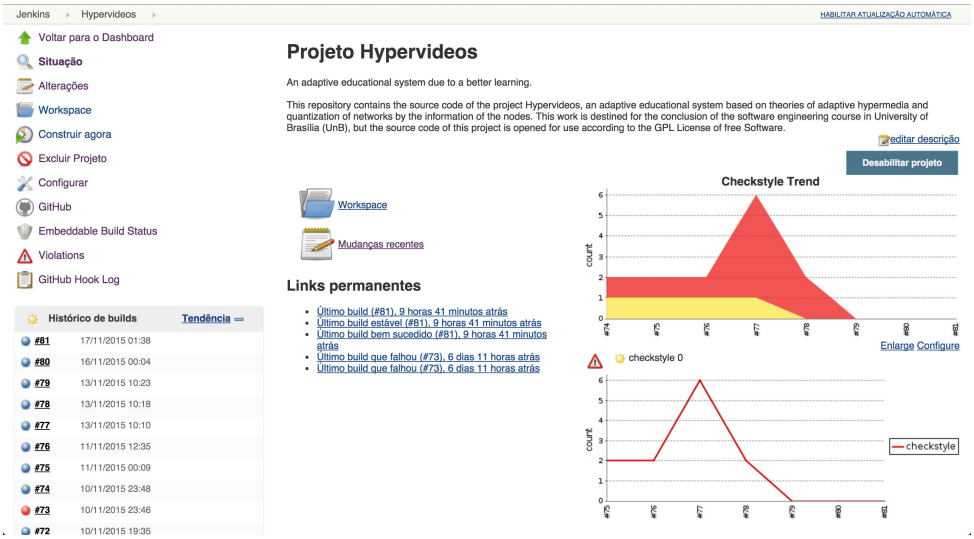
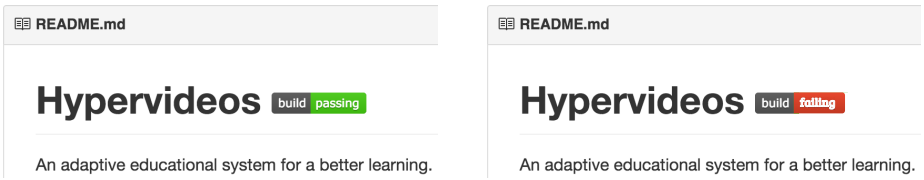


Figura 19 – Ferramenta de Integração Contínua para o projeto Hypervideos.

Os resultados das *builds* podem ser visualizados diretamente no repositório do *Github*, por meio do *plugin* de *status* da última construção do projeto, como mostrado na Figura 20, que apresenta como o indicador aparece quando a construção e testes passam (Fig. 20a), ou não (Fig 20b).



(a) *build* bem sucedida. (b) *build* mal sucedida.

Figura 20 – Status da última construção do projeto no repositório de código.

2.6 Aspectos Gerais

De forma geral, o processo de desenvolvimento do sistema foi guiado por marcos (i.e. *milestones*) definidos no repositório do projeto. Para que o objetivo de cada marco fosse alcançado, foram definidas atividades a serem concluídas, registradas em *issues*, sendo estimadas, implementadas e testadas dentro do período de duração de cada *milestone* do projeto. Para organização do fluxo de trabalho, foi utilizado um *kanban* para gerenciar as *issues* que seriam desenvolvidas a cada semana[1].

[1] Todas as atividades e marcos do projeto podem ser visualizados no repositório no *Github* (<www.github.com/ArthurJahn/hypervideos>)

O foco inicial do trabalho foi levantar um ambiente com as ferramentas necessárias para o desenvolvimento (como os ambientes local e integração contínua). Posteriormente, foi desenvolvido o módulo de autoria, o módulo de visualização e só então o algoritmo de quantização da rede. Entre estas etapas o sistema sofreu várias mudanças, as mais importantes são:

- mudanças na arquitetura, sendo estas: a separação do modelo de domínio; a separação da camada de visão por meio da componentização; e a organização da camada de controle, como ilustrado na Figura 14;
- mudanças dos testes da aplicação, que inicialmente eram apenas testes *Meteor* e passaram a englobar testes *Polymer*;
- mudanças nos *plugins* do ambiente de integração contínua para englobar análise estática de código e disparo automático de *builds*;
- mudanças nos *scripts* de teste para adicionar relatórios de cobertura e de análise de código;
- mudanças no estilo de codificação para adequar ao padrão *JavaScript* segundo o *Meteor*.

Estas mudanças reafirmam a necessidade de adaptação do planejamento e de revisões do trabalho executado, sendo todas decorrentes do fracasso em algum ponto, mas que foram corrigidos e permitiram a continuidade do desenvolvimento. O resultado deste trabalho é apresentado no tópico a seguir.

3 Sistema Adaptativo de Vídeos Interativos

Tendo como base as teorias apresentadas, foi desenvolvido um sistema que utiliza o modelo arquitetural MVC e que contempla duas funcionalidades principais: um módulo para autoria de cursos compostos por vídeos interativos e um módulo para visualização adaptativa destes cursos, que contemple adaptações de conteúdo e navegação. Este capítulo apresenta os resultados obtidos no desenvolvimento, englobando o módulo de autoria de cursos, o módulo de visualização adaptativa e o mecanismo de quantização por meio da QRN.

3.1 Módulo de Autoria de Cursos

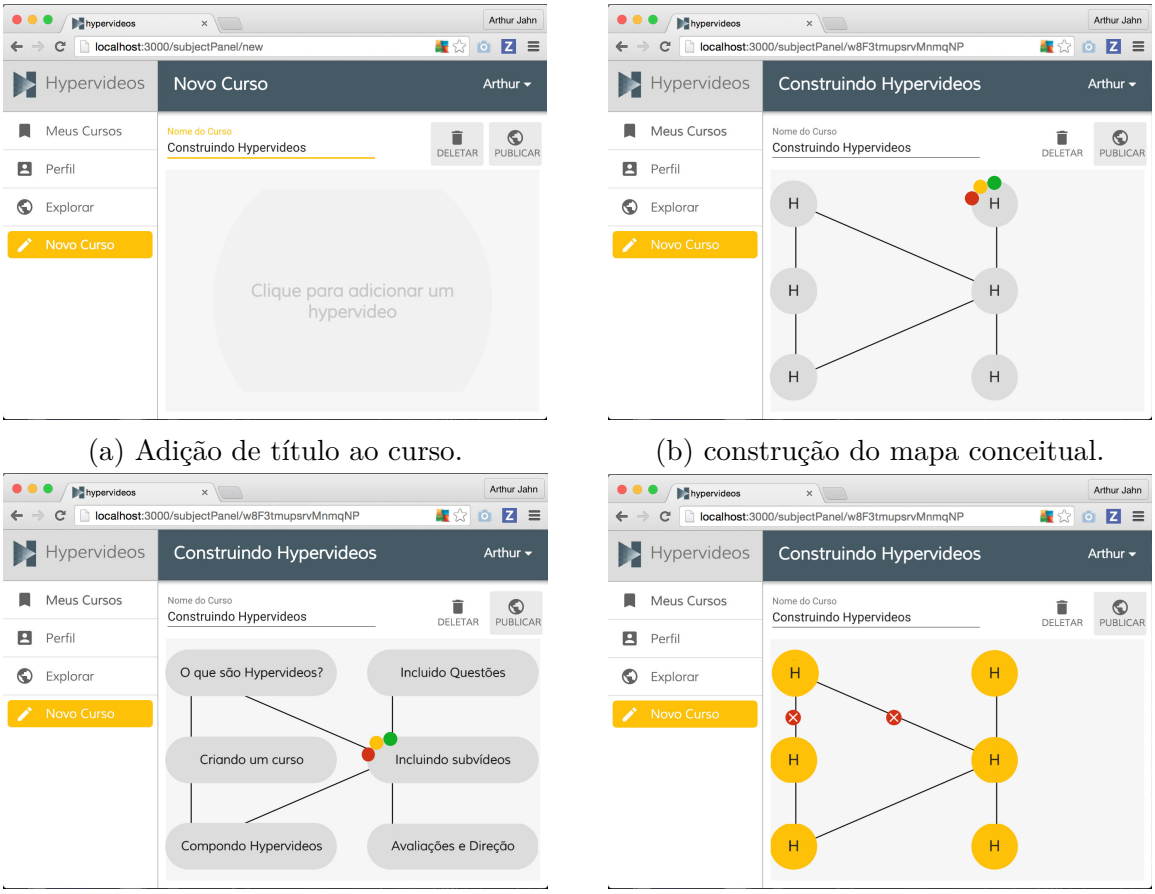
O módulo de autoria de cursos tem o propósito de permitir que educadores construam materiais de estudo para enriquecer a base de dados do sistema proposto. Este módulo pode ser mais facilmente compreendido se dividido em dois níveis: o nível conceitual e o nível de construção do hypervideo.

No nível conceitual, o professor poderá criar um novo curso, seus hipervideos e as ligações entre eles, construindo assim, a rede na qual o estudante navegará. Este nível representa o mapa conceitual que aparecerá para o estudante quando pesquisar um curso no sistema, sendo de extrema importância, pois o aprendizado é mais efetivo se o estudante tiver conhecimento do todo antes das partes.

A Figura 21a mostra o componente de criação de um curso, no qual existe um campo para a definição do título e uma área central que permite a composição do mapa conceitual, por meio da adição, remoção, interligação ou edição de cada hypervideo adicionado durante a construção do curso.

Novos hypervideos podem ser adicionados clicando na área de composição do curso. A Figura 21b ilustra um mapa construído, no qual podem ser visualizados os ícones que representam hypervideos. Clicando em um hypervideo criado, é possível ver o conceito definido para ele, como ilustra a Figura 21c, com todos os conceitos do mapa visíveis. A Figura 21d mostra o mapa conceitual em modo de ligação, para permitir a criação e deleção de *links* entre hypervideos.

É possível visualizar, nas Figuras 21b e 21c, três botões coloridos sobre o ícone de um hypervideo. Esses botões aparecem quando o cursor passa sobre o ícone e permitem que o usuário execute as seguintes ações: remover o hypervervídeo (botão vermelho), criar conexões para o hypervideo (botão amarelo) ou construir o conteúdo do hypervídeo (botão verde), esta última sendo discutida a seguir.



(a) Adição de título ao curso.

(b) construção do mapa conceitual.

(c) visualização dos conceitos.

(d) modo de edição de ligações.

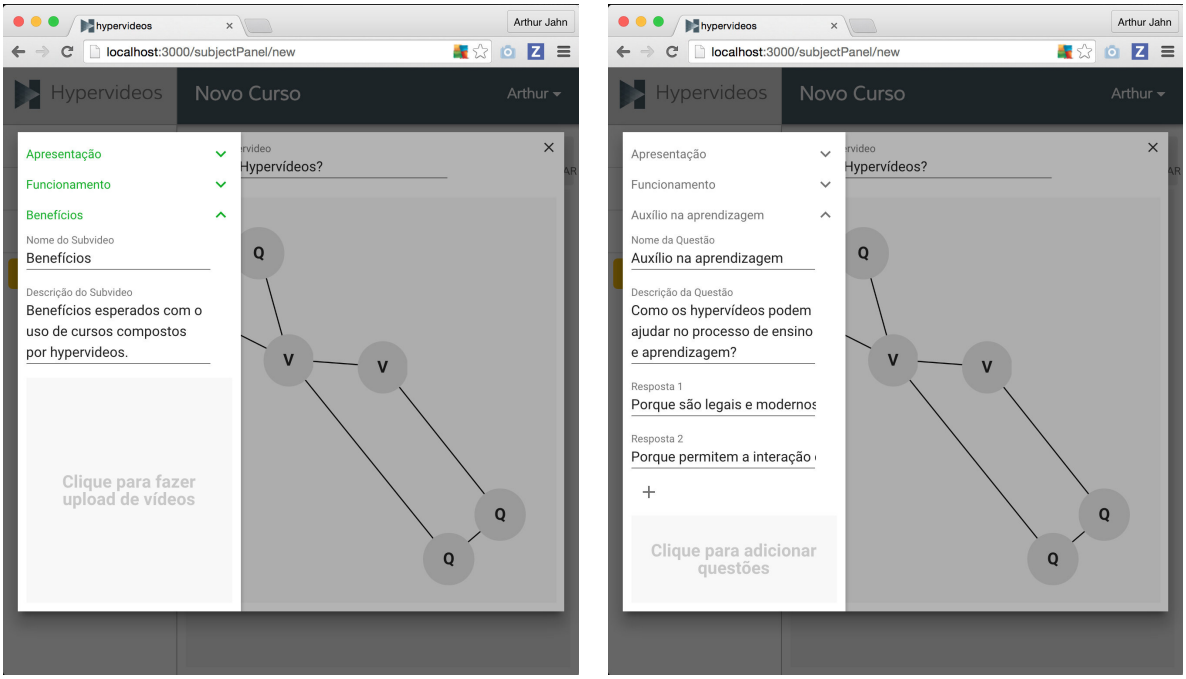
Figura 21 – Funções do módulo de autoria no nível conceitual.

No nível de construção do conteúdo de um hypervideo é possível adicionar subvídeos e questões para criar a navegação no hypervideo.

Subvídeos são criados por meio do carregamento de arquivos de vídeo na aba de vídeos e criam automaticamente questões vinculadas. O sistema permite também que mais questões sejam criadas dentro da aba de questões.

Subvídeos e questões possuem um nível de visibilidade para restringir que materiais muito complexos ou muito simples apareçam para o usuário que assiste ao curso. A Figura 22 apresenta a página de composição de um hypervídeo com três subvídeos (Fig. 22a) e três questões (Fig. 22b). Ao fundo da figura é possível ver o grafo de conexões do hypervideo.

Após a construção de todos os hypervídeos, o autor poderá publicar o curso na rede e a partir desse momento, outros usuários poderão assistir ao material educativo. Essa parte da interação é apresentada no tópico seguinte, a respeito do módulo de visualização adaptativa.



(a) Subvídeos carregados. (b) Questões criadas.

Figura 22 – Funções do módulo de autoria no nível de construção do hypervideo.

3.2 Módulo de Visualização Adaptativa

O módulo de visualização adaptativa permite que o usuário assista um curso disponibilizado na rede de acordo com seu nível no sistema. As operações executadas pelo usuário neste módulo começam no momento da busca por um curso. Na página "Explorar", é possível assistir a um curso ou adicioná-lo à biblioteca para assistir depois, como mostra a Figura 23.

Quando o usuário opta por assistir um curso, ele é direcionado para a página de apresentação adaptativa, como mostra a Figura 24. Nesse ambiente, o usuário é capaz de responder às questões propostas pelo professor, selecionar subvídeos de sua preferência e visualizar seu progresso no fluxo do curso.

O módulo de visualização possui um componente para que o usuário possa fazer anotações sobre o hypervideo, que servem de suporte para o aprendizado do conteúdo apresentado. É importante ressaltar que a adaptação do conteúdo foi alcançada por meio do filtro dos subvídeos e questões segundo o perfil do usuário, que aparecem nas abas ao lado do reprodutor de vídeo, juntamente com o mapa do curso.

3.3 Mecanismo de Quantização por meio da QRN

O mecanismo de quantização permite que a navegação entre hypervideos de um curso seja adaptativa de acordo com os escores obtidos pelo estudante, a confiabilidade

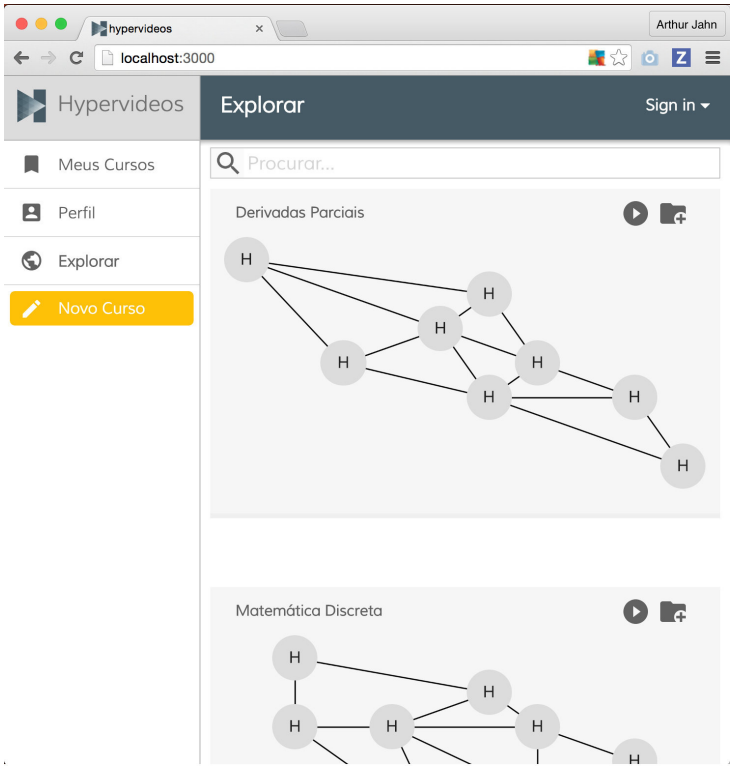


Figura 23 – Página de busca por cursos.

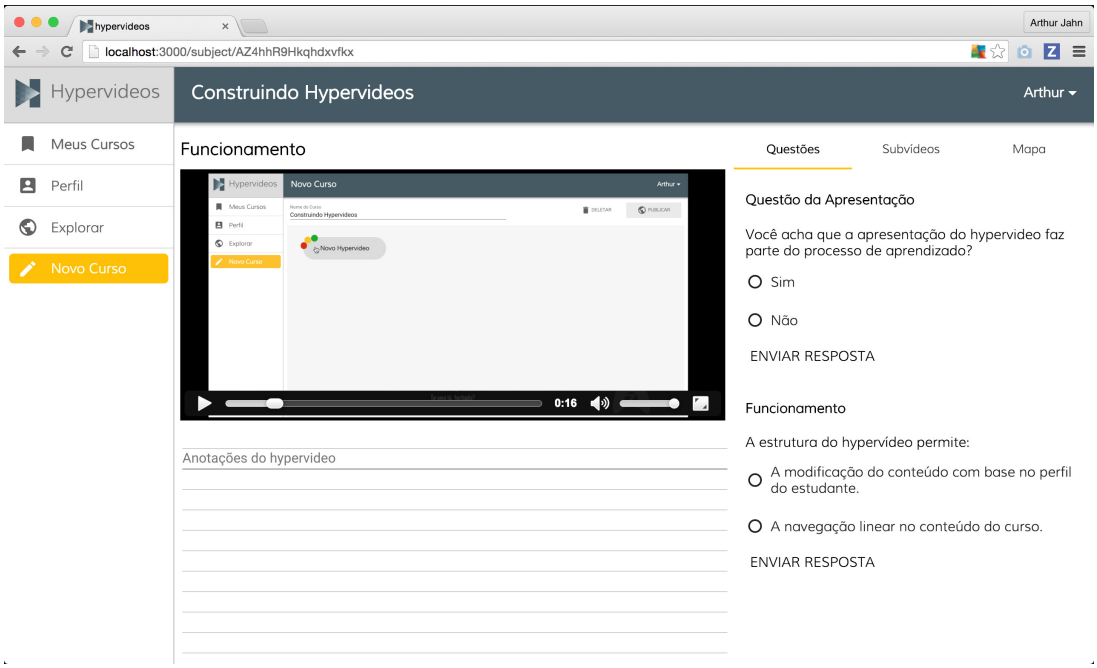


Figura 24 – Módulo de visualização adaptativa.

da avaliação, o decaimento temporal e as ligações que o hypervideo possui. Esse cálculo é feito unicamente no servidor, que envia a resposta das operações para o cliente, permitindo assim, que outros hypervideos sejam indicados pelo sistema.

O *Meteor* permite essas operações por meio de métodos (i.e. *Meteor Methods*) executados unicamente no servidor e simulados no cliente, que é atualizado quando a

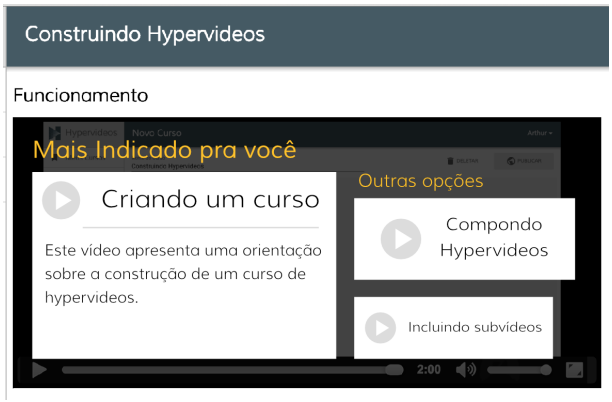


Figura 25 – Componente de apresentação dos hypervideos indicados.

operação no servidor é concluída e devolvida. Na arquitetura do sistema, esses métodos são chamados pelo modelo, o que faz sentido, já que são operações da lógica do domínio da aplicação e compõem a parte do modelo que está definida no servidor. O componente de apresentação dos hypervideos sugeridos pelo sistema aparece como mostrado na Figura 25.

Apesar do componente de apresentação dos hypervideos indicados estar construído, até o momento da escrita deste trabalho não foi possível validar a implementação da QRN com um curso real no sistema, apenas por meio de inspeção e validação das equações utilizadas. Isso devido aos coeficientes para indicar a direção da evolução no curso e também, ao coeficiente relacionado aos blocos coesos, representados por meio das ligações entre hypervideos, que necessita ser calibrado para não impossibilitar que outros nodos da rede sejam acessados. Esta seção apresenta os resultados obtidos no desenvolvimento, englobando o módulo de autoria de cursos, o módulo de visualização adaptativa e o mecanismo de quantização por meio da QRN.

Considerações Finais

Neste capítulo são apresentadas as considerações a respeito da pesquisa executada dos pontos de vista dos objetivos esperados e das oportunidades para trabalhos futuros.

Objetivos da Pesquisa

Após a revisão bibliográfica, foi possível verificar que os estudos sobre Sistemas de Hipermídias Adaptativas e Objetos de Aprendizagem, em especial vídeos interativos, estão bastante avançados. Entretanto, apenas nos últimos anos houve uma aproximação entre as áreas. Além disso, a desmotivação dos professores, o desnível entre os ingressantes e o alto nível de reprovações aparecem como fatores críticos para a desistência de estudantes nos cursos relacionados a engenharia (SILVA, 2005).

A construção de sistemas educativos adaptativos aparece como uma boa proposta para a diminuição do desnível entre estudantes e como ferramenta motivadora para alunos mais avançados, unindo recentes pesquisas sobre objetos de aprendizagem (OA) e sistemas de hipermídias adaptativas (SHA). Porém, é necessário prover mecanismos para que professores possam criar os materiais educativos nesses sistemas.

Este trabalho apresentou as teorias relacionadas ao desenvolvimento de sistemas adaptativos e, em especial, a teoria de quantização de redes por nodos (QRN), que permite a construção de uma navegação global adaptativa para a rede de um curso. Essa teoria embasou o desenvolvimento do sistema, levando em consideração elementos importantes no âmbito da engenharia de *software*, como arquitetura, testes, gerência de configuração e métodos ágeis.

Sob o ponto de vista dos objetivos do trabalho, foi possível desenvolver um módulo de autoria de cursos compostos por vídeos interativos para que professores possam contribuir para a construção de uma base de cursos *online* com o propósito de motivar e melhor nivelar estudantes. Foi desenvolvido também um módulo de visualização adapativa que, no momento, dá suporte apenas à ocultabilidade, mas que permite ao professor utilizar conceitos mais elaborados, como organizadores prévios, diferenciação progressiva ou ainda a reconciliação integrativa definidos por Ausubel (2000). No que diz respeito a navegação global adaptativa, foi possível implementar os algorítmos relacionados à QRN e testá-los unitariamente, verificando o comportamento de cada equação, mas não integralmente, dado que durante o tempo de desenvolvimento do sistema, não se conseguiu calibrar os coeficientes relacionados a direção e blocos coesos.

Trabalhos Futuros

Conforme esperado com o desenvolvimento desta pesquisa, surgiram vários pontos de melhoria da plataforma que não puderam ser adicionados ao desenvolvimento, devido ao escopo e tempo disponíveis. Um deles foi a verificação das teorias relacionadas à composição de vídeos interativos que incluem controle temporal e anotações no momento da construção do material, as teorias dos hipervídeos (SADALLAH; AUBERT; PRIé, 2012).

Devido à organização da arquitetura do sistema, as modificações exigidas para a implementação de estruturas que permitam a utilização dessas teorias não seriam muito grandes e impactariam mais na lógica da visão, em componentes *Polymer*, do que nas camadas do modelo ou do controle.

Ainda com a utilização das teorias de hipervídeos, outros elementos mais complexos definidos por Ausubel (2000) poderiam ser incorporados ao sistema, como os organizadores prévios, o que possibilita a diminuição do esforço do professor para a criação de cursos.

No sentido da adaptação, tanto do conteúdo quanto da navegação, como trabalho futuro é necessário que se defina categorias para os perfis de usuário e também que ocorra a verificação das mudanças sofridas pelo estudante, adequando-o de acordo com o seu avanço no uso do sistema. Para essa finalidade, podem ser utilizadas teorias relacionadas a Sistemas Tutores Inteligentes (STI) que monitorem a evolução dos estudantes na plataforma e que possibilitem a seleção de materiais diferenciados segundo cada perfil, como questões para avaliação ou vídeos com abordagens de ensino diferentes (FRAGELLI, 2010).

Além dos fatores relacionados às teorias de aprendizagem estudadas, existem as questões da engenharia de software aplicadas à plataforma desenvolvida. O propósito do desenvolvimento foi prover uma arquitetura estável que fosse passível de atualizações na visão sem que o modelo ou o controle fossem modificados, já que novas teorias de apresentação de conteúdo podem vir a substituir a proposta desenvolvida até o momento.

Para tal fim, existem ainda aspectos que devem ser modificados para melhorar a arquitetura e prover novos recursos, como a combinação dos componentes visuais *Polymer* no momento de *deploy* da aplicação (i.e. vulcanização), que pode reduzir significativamente o tamanho dos arquivos e o número de requisições ao servidor no carregamento inicial da aplicação (OSMANI, 2013). Essa modificação implica no ambiente de integração contínua, que poderá executar o processo de *build* e teste com menos memória.

Outro ponto existente é o gerenciamento de pacotes e dependências *npm*[1] para a arquitetura proposta, já que o *Polymer* e algumas outras ferramentas não são nativamente suportados pelo *Meteor*, o que deve exigir a inclusão de um gerenciador de dependências

[1] Ferramenta de gerenciamento de pacotes Node.js. Mais informações em: <https://www.npmjs.com/>

npm, como exemplo o *meteor-npm* [2]. Nesse mesmo ponto, ainda não existe um *script* para configuração do ambiente de desenvolvimento completo, apenas o tutorial na wiki do repositório.

Apesar do cumprimento dos objetivos e das oportunidades de evolução da plataforma desenvolvida, o aprendizado adquirido com este trabalho e a possibilidade de atuar ativamente no processo ensino-aprendizagem, contribuindo para a formação de melhores profissionais, foram os principais resultados obtidos. Espera-se, com esta plataforma, que novas evoluções venham a ocorrer para aprimorar e colocar em prática as teorias estudadas e aqui aplicadas.

[2] Ferramenta de gerenciamento de pacotes *npm* compatível com *plugins Meteor*. Mais informações em: <https://www.npmjs.com/package/meteor-npm>

Referências

AGILE ALLIANCE. *Continous Integration*. [S.l.], 2015. Disponível em: <http://guide.agilealliance.org/guide/ci.html>. Citado na página 49.

AUSUBEL, D. *The Acquisition and Retention of Knowledge: A Cognitive View*. Springer Netherlands, 2000. ISBN 9780792365051. Disponível em: <https://books.google.com.br/books?id=cwV_1uIpgVAC>. Citado 5 vezes nas páginas 30, 31, 32, 69 e 70.

BAKER, J. *Internet: systems and applications*. EMCParadigm, 2006. ISBN 9780763822590. Disponível em: <https://books.google.com.br/books?id=33_7usX-wvQC>. Citado na página 44.

BOLLEN, J.; HEYLIGHEN, F. Algorithms for the self organization of distributed, multi-user networks. possible application to the future world wide web. *Cybernetics and Systems '96*, World Science, 1996. Citado na página 36.

BRUSILOVSKY, P. *The Construction and Application of Student Models in Intelligent Tutoring Systems**. 1994. Citado 2 vezes nas páginas 25 e 32.

BRUSILOVSKY, P. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, Kluwer Academic Publishers, v. 6, n. 2-3, p. 87–129, 1996. ISSN 0924-1868. Disponível em: <http://dx.doi.org/10.1007/BF00143964>. Citado 5 vezes nas páginas 25, 32, 34, 35 e 36.

BRUSILOVSKY, P. Adaptive navigation support in educational hypermedia: The role of student knowledge level and the case for meta-adaptation. British Journal of Educational Technology, 2003. Citado na página 34.

CASCIARO, M. *Node.js Design Patterns*. Packt Publishing, 2014. (Community Experience Distilled). ISBN 9781783287321. Disponível em: <https://books.google.com.br/books?id=Ys4GBgAAQBAJ>. Citado na página 55.

CHANDLER, P. The crucial role of cognitive processes in the design of dynamic visualizations. Learning and Instruction, 2004. Citado na página 33.

CLARK, R. C.; MAYER, R. E. *E-learning and the Science of Instruction*. 3. ed. São Francisco, USA: Pfeiffer, 2011. Citado 2 vezes nas páginas 31 e 32.

COMMITTEE, I. C. S. S. C. et al. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE, 1990. (IEEE Std). ISBN 9781559370677. Disponível em: <https://books.google.com.br/books?id=qGR_PgAACAAJ>. Citado na página 47.

FRAGELLI, R. R. *Uma Abordagem de Redes Quantizadas e Objetos Multiformes para Modelagem de Domínio em Sistemas de Tutoria Inteligentes*. 163 p. Tese (Doutorado), Brasília, Brasil, 2010. Citado 12 vezes nas páginas 25, 26, 29, 30, 31, 32, 34, 35, 36, 37, 40 e 70.

FRIGOTTO, G. *A produtividade da escola improdutiva: um (re)exame das relações entre educação e estrutura econômica social e capitalista*. [S.l.]: Cortez, 1989. 235 p. Citado na página 25.

GAUDREAU, S.; CHAN, M. *Video Interactive Learning System (VILS)*. [S.l.]: Elsevier, 1984. Citado 2 vezes nas páginas 25 e 33.

GIRAFFA, L. M. M. Fundamentos de teorias de ensino-aprendizagem e sua aplicação em sistemas tutores inteligentes. *Instituto de Informática-UFRGS. Porto Alegre*, 1995. Citado na página 25.

GITHUB, INC. *Where Software is Built*. 2015. Disponível em: <https://github.com/>. Citado na página 60.

GOOGLE, i. Mvc architecture. 2015. Disponível em: <https://developer.chrome.com/apps/app_frameworks>. Citado na página 45.

GRAILS ORG. *The Grails Framework*. 2015. Disponível em: <https://grails.org/>. Citado na página 53.

HILL, W. Learning: A survey of psychological interpretation. Allyn and Bacon, Boston, USA, 2002. Citado na página 29.

IEEE. *SWEBOK Guide to the Software Engineering Book of Knowledge*. 2014. Disponível em: <https://luiscastellanos.files.wordpress.com/2007/03/swebokv3.pdf>. Citado na página 59.

JADE TEAM. *JAVA Agent Development Framework*. 2015. Disponível em: <http://jade.tilab.com/>. Citado na página 54.

JENKINS CI. *Jenkins: An Extensible Open Source Continuous Integration Server*. 2015. Disponível em: <https://jenkins-ci.org/>. Citado na página 60.

JUNIT. *JUnit testing framework*. [S.l.], 2015. Disponível em: <http://junit.org/>. Citado na página 47.

KADIRA ORG. *Kadira - Performance Management Platform for Meteor*. 2015. Disponível em: <https://kadira.io/platform>. Citado na página 59.

KRASNER, G. E.; POPE, S. T. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming*, n. 3, p. 26–49, 1988. Disponível em: <http://citeseer.nj.nec.com/krasner88description.html>. Citado 3 vezes nas páginas 43, 44 e 46.

MATLIN, M. *Psicologia cognitiva*. LTC, 2004. ISBN 9788521613923. Disponível em: <https://books.google.com.br/books?id=FzGvAwAACAAJ>. Citado na página 30.

MAYER, R. E. *Multimedia Learning*. Cambridge University Press, 2014. ISBN 9781139129879. Disponível em: <https://books.google.com.br/books?id=4ZbgoQEACAAJ>. Citado 2 vezes nas páginas 32 e 34.

MAYER, R. E.; CHANDLER, P. When learning is just a click away: Does simple user interaction foster deeper understanding of multimedia messages? Journal of Educational Psychology, 2001. Citado 3 vezes nas páginas 31, 32 e 33.

MEJIA, A. Ruby on rails architectural design. 2011. Disponível em: <http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/>. Citado na página 45.

METEOR ORG. *The JavaScript App Platform*. 2015. Disponível em: <https://meteor.com/>. Citado na página 53.

METEOR ORG. *Meteor Blaze*. 2015. Disponível em: <https://www.meteor.com/blaze>. Citado na página 56.

MOCHA WEB. *Node.js Mocha tests*. [S.l.], 2015. Disponível em: <https://mochajs.org/>. Citado 2 vezes nas páginas 47 e 58.

MONGODB. *Meteor: Build iOS and Android Apps that are a Delight to Use*. 2015. Disponível em: <https://www.mongodb.com/blog/post/meteor-build-ios-and-android-apps-are-delight-use>. Citado na página 55.

MOREIRA, M. A. *Teorias de Aprendizagem*. São Paulo, Brasil: EPU, 1999. Citado 2 vezes nas páginas 29 e 30.

MORENO, R.; MAYER, R. E. A learner-centered approach to multimedia explanations: Deriving instructional design principles from cognitive theory. Carolina do Norte, USA, 2000. Citado 3 vezes nas páginas 31, 32 e 33.

NAIK, K.; TRIPATHY, P. *Software Testing and Quality Assurance Theory and Practice*. [s.n.], 2008. Disponível em: <http://goo.gl/jAACG7>. Citado 2 vezes nas páginas 47 e 48.

NETO, A. C. D. Introdução a teste de sofware. 2005. Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>. Citado na página 46.

NODE FOUNDATION. *Node Js*. 2015. Disponível em: <https://nodejs.org/>. Citado na página 54.

NOVAK, J. D.; CAÑAS, A. J. The origins of the concept mapping tool and the continuing evolution of the tool. *Information Visualization*, SAGE Publications, v. 5, n. 3, p. 175–184, 2006. Citado na página 31.

OLIVEIRA, R. A possibilidade da escola unitária na sociedade capitalista. *Cadernos de Educação*, FaE/PPGE/UFPel, 2009. Citado na página 25.

OSMANI, A. *Concatenating Web Components with Vulcanize: Reduce network requests by flattening HTML Imports*. 2013. Disponível em: <https://www.polymer-project.org/0.5/articles/concatenating-web-components.html>. Citado na página 70.

PAIVIO, A. Mental representation: A dual coding approach. Oxford University Press, Oxford, England, 1986. Citado na página 31.

PALAZZO, L. A. M. *Modelos Proativos para Hipermidia Adaptativa*. Tese (Doutorado), Rio Grande do Sul, Brasil, 2000. Citado 3 vezes nas páginas 35, 36 e 37.

PETITIT, F.; TRICOT, M. The new web application architectures and their impacts for enterprises. 2014. Disponível em: <http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/>. Citado na página 45.

POLYMER AUTHORS. *Polymer: Production ready*. 2015. Disponível em: <https://www.polymer-project.org/1.0/>. Citado 2 vezes nas páginas 55 e 56.

QUINN, C. Meteor: bringing the awesome back to web dev. 2015. Disponível em: <https://www.gravitywell.co.uk/latest/how-to/posts/meteor-bring-the-awesome-part-0/>. Citado 2 vezes nas páginas 45 e 46.

REENSKAUG, T. Models - views - controllers. 1979. Disponível em: <https://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>. Citado na página 44.

ROBINS, R. W.; GOSLING, S. D.; CRAIK, K. H. An empirical analysis of trends in psychology. *American Psychologist*, American Psychological Association, v. 54, n. 2, p. 117, 1999. Citado na página 30.

RUBY ON RAILS. *A Guide to Testing Rails Applications*. [S.l.], 2015. Disponível em: <http://guides.rubyonrails.org/testing.html>. Citado na página 47.

RUBY ON RAILS. *Web Development that Doesn't Hurt*. 2015. Disponível em: <http://rubyonrails.org/>. Citado na página 53.

SADALLAH, M.; AUBERT, O.; PRIé, Y. Chm: an annotation- and component-based hypervideo model for the web. Kluwer Academic Publishers, France, 2012. Citado na página 70.

SCHWABER, K.; SUTHERLAND, J. The definitive guide to scrum: The rules of the game. 2013. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>. Citado na página 48.

SHABAITAH, A. R. *Server-Based Desktop Virtualization*. Dissertação (Mestrado), 2014. Citado na página 44.

SHAW, M.; GARLAN, D. *Software architecture: perspectives on an emerging discipline*. [S.l.]: Prentice Hall Englewood Cliffs, 1996. v. 1. Citado na página 43.

SILVA, J. E. Um berço para o homem e o legado skinneriano na educação: do behaviorismo a um novo paradigma para a sociedade do conhecimento. Instituto Politécnico da Guarda, 2005. Citado 4 vezes nas páginas 26, 29, 30 e 69.

STURZBECHER, A. J. et al. Um sistema de autoria para vídeos interativos. PAEE 2013, 2013. Citado 2 vezes nas páginas 32 e 33.

SUTHERLAND, J. *A Arte de Fazer o Dobro do Trabalho na Metade do Tempo*. [S.l.]: Leya, 2014. Citado 2 vezes nas páginas 48 e 49.

TAVARES, R. Aprendizagem significativa, codificação dual e objetos de aprendizagem. Revista Brasileira de Informática na Educação, Paraíba, Brasil, 2010. Citado na página 31.

UBL, M. Tradeoffs in server side and client side rendering. 2015. Disponível em: <https://medium.com/google-developers/tradeoffs-in-server-side-and-client-side-rendering-14dad8d4ff8b>. Citado na página 46.

VICARI, R. M.; GIRAFFA, L. M. M. Fundamentos dos sistemas tutores inteligentes. *Barone, D.; et alii. Sociedades artificiais: a nova fronteira da inteligência nas máquinas. Porto Alegre: Bookman*, 2003. Citado na página 25.

WETZEL, C.; RADTKE, P.; STERN, H. *Instructional effectiveness of video media*. Lawrence Erlbaum Associates, 1994. ISBN 9780805816983. Disponível em: <https://books.google.com.br/books?id=qGbuAAAAMAAJ>. Citado na página 33.

XOLVIO. *Getting started with Meteor Cucumber*. 2015. Disponível em: <https://chimp.readme.io/docs/getting-started-with-meteor-cucumber>. Citado na página 58.

XOLVIO. *Jasmine tests*. 2015. Disponível em: <http://jasmine.github.io/2.3/introduction.html>. Citado na página 58.

ZHANG, D. Interactive multimedia-based e-learning: A study of effectiveness. *American Journal of Distance Education*, v. 19, n. 3, p. 149–162, 2005. Disponível em: <http://dx.doi.org/10.1207/s15389286ajde1903_3>. Citado 2 vezes nas páginas 25 e 33.

Apêndices

APÊNDICE A – Suporte Tecnológico

Este apêndice descreve todas as ferramentas utilizadas ao longo deste trabalho.

A.1 Ferramentas de Suporte ao Desenvolvimento

- **Homebrew:** Gerenciador de pacotes para ambientes OS X - versão 0.9.5.
- **Curl:** Aplicação para transferência de dados com syntaxe de URL. Permite a instalação de pacotes e ferramentas - versão 7.43.0.
- **Node.js:** Aplicação para construção de sistemas *web*, possui vários utilitários, e é necessário para instalação do *npm* - versão 5.2.0.
- **npm:** *Node Package Manager* utilizado para gerenciar não só pacotes do node, mas também pacotes de várias aplicações *JavaScript* 3.3.12.
- **MongoDB:** Sistema gerenciador de banco de dados orientado a documentos, sendo utilizado como SGBD padrão para aplicações *Meteor* - versão 3.2.0.
- **Meteor:** *Framework* para desenvolvimento de aplicativos e sistemas *web* reativos - versão 1.2.
- **Polymer:** Ferramenta utilizada para a componentização da interface da aplicação - versão 1.1.5.
- **Bower:** Ferramenta utilizada para gerenciamento de componentes *Polymer* - versão 1.7.5.
- **Velocity:** *Framework* de testes para *Meteor* - versão 1.1.0.
- **ESlint:** Ferramenta para análise estática de código *JavaScript* - versão 1.10.3.
- **Kadira:** Ferramenta para análise de desempenho para aplicações *Meteor* - versão 0.8.2.
- **Mocha:** Motor de testes para aplicações *JavaScript* - versão 0.6.4.
- **Astronomy:** Biblioteca para separação da camada de modelo para aplicações *Meteor* - versão 1.2.
- **Accounts-password:** Biblioteca para gerenciamento de contas de usuários para *Meteor* - versão 1.1.4.

- **Web-component-tester:** Ferramenta de testes unitários para componentes *Polymer* - versão 4.0.3.
- **Web-component-tester-istanbul:** Gerador de relatório de cobertura de código para componentes *Polymer* 0.10.0.
- **cfs-standard-packages:** Pacotes *Meteor* para gerenciamento de arquivos de mídia, no caso, vídeos - versão 0.0.30.
- **cfs-gridfs:** Pacote *Meteor* para persistência de arquivos de mídia em um banco GridFS MongoDB - versão 0.0.12.
- **Search Source:** Pacote para facilitar a criação de *queries* de busca reativas em aplicações *Meteor* - versão 1.4.2.
- **Iron-router:** Pacote de gerenciamento de rotas para aplicações *Meteor* - versão 1.0.12.
- **publish-composite:** Pacote para criação de composições em *queries* recursivas para aplicações *Meteor* - versão 1.4.2.

A.2 Ferramentas para Ambiente de Integração contínua

O ambiente de integração contínua utilizou todas as ferramentas necessárias no desenvolvimento, além das seguintes:

- **Jenkins:** Ambiente de Integração Contínua - versão 1.642.
- **Jenkins Git Plugin:** Permite que operações do git possam ser feitas no *script* de Integração contínua - versão 2.4.0.
- **Jenkins Github Plugin:** Permite vincular *jobs* do jenkins com repositórios no *Github* - versão 1.13.3.
- **Jenkins Checkstyle Plugin:** Permite o acompanhamento de relatórios de aderência do código aos padrões - versão 3.43.
- **Jenkins Violations Plugin:** Gera gráficos e indicadores de qualidade de código - versão 0.7.11.
- **Jenkins External Monitor Job Type Plugin:** Permite que *jobs* do Jenkins sejam disparados por operações externas, como *pushes* no repositório do projeto - versão 1.4.
- **Jenkins Embeddable Build Status:** Permite que o status atual da *build* do projeto seja vinculado a qualquer página externa - versão 1.8.